Syracuse University

SURFACE

Electrical Engineering and Computer Science College of Engineering and Computer Science

2008

Mitigating DoS attacks against broadcast authentication in wireless sensor networks

Peng Ning
North Carolina State University at Raleigh

An Liu
North Carolina State University

Wenliang Du
Syracuse University, wedu@syr.edu

Follow this and additional works at: https://surface.syr.edu/eecs

 Part of the Computer Sciences Commons

Recommended Citation

Ning, Peng; Liu, An; and Du, Wenliang, "Mitigating DoS attacks against broadcast authentication in wireless sensor networks" (2008). *Electrical Engineering and Computer Science*. 105.
https://surface.syr.edu/eecs/105

This Article is brought to you for free and open access by the College of Engineering and Computer Science at SURFACE. It has been accepted for inclusion in Electrical Engineering and Computer Science by an authorized administrator of SURFACE. For more information, please contact surface@syr.edu.

Mitigating DoS Attacks against Broadcast Authentication in Wireless Sensor Networks

Peng Ning, An Liu
North Carolina State University
and
Wenliang Du
Syracuse University

Broadcast authentication is a critical security service in wireless sensor networks. There are two general approaches for broadcast authentication in wireless sensor networks: digital signatures and μTESLA-based techniques. However, both signature-based and μTESLA-based broadcast authentication are vulnerable to Denial of Services (DoS) attacks: An attacker can inject bogus broadcast packets to force sensor nodes to perform expensive signature verifications (in case of signature-based broadcast authentication) or packet forwarding (in case of μTESLA-based broadcast authentication), thus exhausting their limited battery power. This paper presents an efficient mechanism called *message specific puzzle* to mitigate such DoS attacks. In addition to signature-based or μTESLA-based broadcast authentication, this approach adds a weak authenticator in each broadcast packet, which can be efficiently verified by a regular sensor node, but takes a computationally powerful attacker a substantial amount of time to forge. Upon receiving a broadcast packet, each sensor node first verifies the weak authenticator, and performs the expensive signature verification (in signature-based broadcast authentication) or packet forwarding (in μTESLA-based broadcast authentication) only when the weak authenticator is valid. A weak authenticator cannot be pre-computed without a non-reusable (or short-lived) key disclosed only in a valid packet. Even if an attacker has intensive computational resources to forge one or more weak authenticators, it is difficult to reuse these forged weak authenticators. Thus, this weak authentication mechanism substantially increases the difficulty of launching successful DoS attacks against signature-based or μTESLA-based broadcast authentication. A limitation of this approach is that it requires a powerful sender and introduces sender-side delay. This paper also reports an implementation of the proposed techniques on TinyOS, as well as initial experimental evaluation in a network of MICAz motes.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*

General Terms: Security, Design, Algorithms

Additional Key Words and Phrases: Sensor Networks, Security, Broadcast Authentication, DoS Attacks

This work is supported by the National Science Foundation (NSF) under grants CAREER-0447761, CNS-0430223 and CNS-0430252, and by the Army Research Office (ARO) under grant W911NF-05-1-0247. The contents of this paper do not necessarily reflect the position or the policies of the U.S. Government.

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 20 ACM 0000-0000/20/0000-0001 $5.00

1. INTRODUCTION

Recent technological advances have made it possible to deploy large scale sensor networks consisting of a large number of low-cost, low-power, and multi-functional sensor nodes that communicate in short distances through wireless links [Akyildiz et al. 2002]. These sensor nodes are typically battery-powered, and are expected to run in an unattended fashion for a long period of time. Such sensor networks have a wide range of applications in civilian and military operations such as monitoring of critical infrastructure and battlefield surveillance. Many attempts have been made to develop protocols that can fulfill the requirements of these applications (e.g., [Perrig et al. 2001; Hill et al. 2000; Niculescu and Nath 2001; Gay et al. 2003; Newsome and Song 2003; Akyildiz et al. 2002]).

Broadcast is an important communication primitive in wireless sensor networks. It is highly desirable to broadcast commands (e.g., queries used to collect sensor data) and data (e.g., global clock value distributed for time synchronization) to the sensor nodes due to the large number of sensor nodes and the broadcast nature of wireless communication. Due to the limited signal range, it is usually necessary to have some receivers of a broadcast packet forward it in order to propagate the packet throughout the network (e.g., through flooding, or probabilistic broadcasting [Ni et al. 1999; Stojmenovic et al. 2002; Levis et al. 2004]). As illustrated in Figure 1, node *S* first broadcasts a packet (locally within the signal range), and *some* nodes that receive this packet for the first time (e.g., node *A*) forward it (through a local re-broadcast) to propagate this packet to more nodes (e.g., node *B*). This process continues until all the reachable nodes receive the broadcast packet.

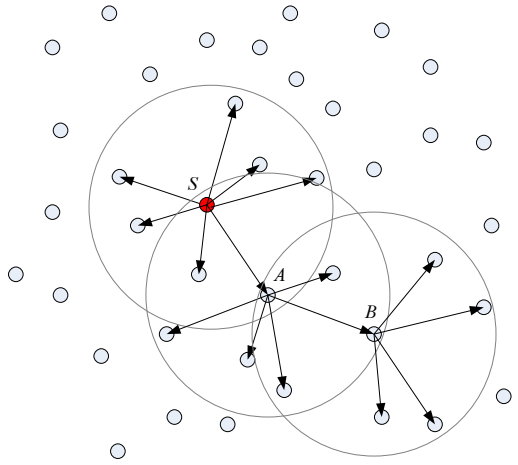

Fig. 1. Broadcast in wireless sensor networks. A broadcast packet is usually forwarded multiple times before all reachable nodes receive it.

For clarify, we refer to the node that originally generates the broadcast packet as the *sender*, and a node that receives a broadcast packet as a *receiver*. As discussed earlier, a receiver may forward the received packet.

1.1 Broadcast Authentication in Sensor Networks

In hostile environments, broadcast authentication (i.e., authentication of broadcast packets) is a critical security service to ensure the trustworthiness of sensor network applications. Due to the resource constraints on sensor nodes (especially the limited battery power) and possible node compromises, broadcast authentication in wireless sensor networks is by no means a trivial problem.

There are two general approaches for broadcast authentication in sensor networks: *digital signatures* and *μTESLA-based approaches*. Public key based digital signatures were initially considered impractical for resource constrained sensor networks. However, it was recently demonstrated that it is feasible to perform public key cryptographic operations on low-end sensor nodes [Gura et al. 2004]. For example, it takes about 0.81 seconds to perform a point multiplication on a 160-bit elliptic curve on Atmega128, the processor used in many sensor nodes such as MICA2 and MICAz motes [Gura et al. 2004]. This implies that it would take about 1.62 seconds to verify an ECDSA signature on the same elliptic curve (or less if optimization for signature verification is used), since the dominant operations in signature verification are two point multiplications. More powerful and energy-efficient sensor platforms (e.g., Intel iMotes [Intel Research]) are being developed, which are expected to perform public key cryptographic operations more quickly. Meanwhile, the recent advances in sensor wireless communication allow relatively large packets to be transmitted. In particular, IEEE 802.15.4, the standard for low-power sensor networks, allows a variable payload of up to 102 bytes [IEEE Computer Society 2003]. Such a packet provides enough space to include a digital signature for broadcast authentication, such as a 40-byte ECDSA signature on the above 160-bit elliptic curve. Thus, it is possible to achieve broadcast authentication with digital signatures in wireless sensor networks.

μTESLA and several of its variations [Perrig et al. 2001; Liu and Ning 2003; 2004; Liu et al. 2005] have been developed in the past several years for scalable broadcast authentication in wireless sensor networks. All of these approaches are based on TESLA [Perrig et al. 2000; 2001], which provides broadcast authentication based on symmetric cryptography by delayed disclosure of authentication keys. (A brief overview of μTESLA can be found in Appendix A.) Compared with digital signatures, μTESLA-based approaches are much more efficient and less resource consuming, but cannot provide authentication immediately after broadcast packets are received.

Both digital signatures and μTESLA-based approaches are vulnerable to Denial of Service (DoS) attacks. This is a fatal threat to sensor networks because of the limited and depletable battery power on sensor nodes.

1.1.1 *DoS Attacks against Signature-Based Broadcast Authentication.* Although it is possible to perform digital signature operations on sensor nodes, the cost of such operations is still substantially higher than that of symmetric cryptographic operations, and will substantially consume the battery power if frequently performed. This leads to a fatal threat to signature-based broadcast authentication: An attacker may simply forge a large number of broadcast messages with digital signatures, force sensor nodes to verify these signatures, and eventually deplete their battery power. Benign sensor nodes may certainly decide not to forward broadcast messages before their signatures are verified. However, a single malicious node can still overload and disable many benign nodes in its local region with forged messages. Moreover, an attacker may generate much higher impact by increasing the signal strength or deploying multiple malicious nodes in different locations.

To see more clearly the impact of forged signature packets on sensor energy consumption, let us take a closer look at the energy consumption for receiving a bogus packet, using MICAz, a typical sensor platform, as an example. A bogus packet sent by the DoS attacker can consume the receiver's energy in at least two steps (assuming bogus packets are not re-broadcast): (1) Receiving the packet; (2) Processing the packet and verifying the signature. Assume the maximum payload of 102 bytes, as defined in IEEE 802.15.4 [IEEE Computer Society 2003]. According to [CC2 2006], for a packet of 102 byte payload, a MICAz mote needs to transmit up to 133 bytes in the physical layer, including preamble sequence, physical layer header, MAC layer header, frame check sequence, and the payload data. Based on the MICAz data sheet [MIC], which gives the receiving current draw 19.7mA, the 250kbps transmit data rate and the 3.0V power level, we can estimate that the receiving energy cost of such a packet at the radio module is at most $3.0 \times 19.7 \times 133 \times 8/250,000 = 0.25$mJ. Now let us see the energy cost of verifying an ECDSA signature. Assume the verification of an ECDSA signature takes 1.62 seconds. In active mode, the current draw of MICAz is 8 mA [MIC]. Thus, the energy cost for a signature verification can be estimated as $3.0 \times 8 \times 1.62 = 38.88$mJ. These numbers show that signature verification consumes much more energy than receiving the same packet. Thus, it is necessary to develop techniques to reduce the number of false signature verifications.

1.1.2 *DoS Attacks against μTESLA-Based Broadcast Authentication.* A major limitation of μTESLA [Perrig et al. 2001] and its variations [Liu and Ning 2003; 2004] is the authentication delay. In other words, a receiver cannot authenticate a broadcast packet immediately after receiving it. Note that a broadcast packet typically has to be forwarded (via local re-broadcast) multiple times before it reaches all the nodes. This means that a sensor node has to forward a broadcast packet before properly authenticating it. The key disclosed in a broadcast packet can provide some weak authentication. However, once an attacker receives a broadcast packet, he/she can reuse this key to forge many packets that can pass this weak authentication. As a result, similar to the DoS attacks against signature-based broadcast authentication, an attacker can force regular nodes to forward a large number of bogus packets to eventually exhaust their battery power.

An immediate authentication mechanism was developed in [Perrig et al. 2001] to partially address this problem. Specifically, a hash image of the *content* of each packet is also included in an earlier packet [Perrig et al. 2001]. Thus, as long as the earlier packet is authenticated, the content of the later packet can be immediately authenticated upon receipt. However, this immediate authentication does not cover the hash image of the later packet content, nor the message authentication code (MAC) in the packet just received. Thus, an attacker can still forge a large number of packets by modifying these uncovered parts without being immediately detected, resulting in the same DoS attack.

1.2 Proposed Approach

In this paper, we develop an approach to mitigate the DoS attacks against both signature-based and μTESLA-based broadcast authentication. The basic idea is to use an efficiently verifiable weak authenticator along with broadcast authentication, so that a sensor node performs the expensive signature verification (in case of signature-based broadcast authentication) or packet forwarding (in case of μTESLA or its variations) only when the weak authenticator can be verified.

We develop a weak authentication mechanism called *message specific puzzle* to achieve

this goal. This mechanism has a number of nice properties:

- The weak authentication mechanism is independent of the broadcast authentication mechanism; it works with both digital signatures and μTESLA (or its variations).
- A weak authenticator can be efficiently verified by a regular sensor node; however, it takes a computationally powerful attacker a substantial amount of time to forge a valid weak authenticator.
- A weak authenticator cannot be pre-computed without a non-reusable (or short-lived) key disclosed only in a valid broadcast packet, and an attacker has very limited time to forge (expensive) weak authenticators after seeing a valid key. Thus, it is difficult for the attacker to forge usable weak authenticators.
- Even if an attacker has significant computational resources to forge one or more weak authenticators, it is difficult to reuse these forged weak authenticators. Thus, this weak authentication mechanism substantially increases the difficulty of launching successful DoS attacks against broadcast authentication.
- The weak authentication mechanism has reasonable communication overhead. For example, when 64-bit keys, message specific puzzles with strength $l = 22$ (which are reasonably strong puzzles), 16-bit packet indexes are used, this approach introduces 14 bytes overhead per packet for signature-based broadcast authentication (compared with, e.g., a 40-byte ECDSA signature), and 6 bytes overhead per packet for μTESLA-based broadcast authentication (compared with an 8-byte key and an 8-byte MAC used by μTESLA).

These desirable properties come with a cost. First, the proposed message specific puzzles require a computationally powerful sender with sufficient power supply. Second, the generation of weak authenticators introduces a delay at the sender. The first issue is in general not a problem; it is certainly acceptable to have one or several laptop senders in a typical sensor network deployment. The second issue is tolerable unless there is a real-time requirement for the broadcast messages. For example, it is generally acceptable to delay for a few minutes before disseminating a new task or a program image to all the sensor nodes. Considering the benefits brought by message specific puzzles, we believe the proposed techniques are useful and practical in wireless sensor networks.

We have implemented the proposed techniques on TinyOS, an operating system for networked sensors, and evaluated them using a network of MICAz motes. Our results indicate that the proposed techniques reasonably increase the program size on sensor nodes (e.g., 1,317 bytes in ROM and 289 bytes in RAM for signature-based broadcast authentication). We also confirm that the proposed weak authentication can be efficiently verified on regular sensor receivers (e.g., 14.6 ms to verify a weak authenticator on a MICAz mote). On the sender side, the proposed techniques introduce tolerable delay (e.g., about 10 seconds delay for reasonably strong message specific puzzles). In general, the experiments demonstrate that the proposed techniques are useful on the current PC (as sender) and MICAz (as receivers) platforms.

1.3 Organization

The remainder of this paper is organized as follows. The next section describes the assumptions of the proposed techniques. Section 3 presents the proposed message specific puzzle weak authentication mechanism, and discusses how to integrate message specific

puzzles with signature-based and μTESLA-based broadcast authentication, respectively. Section 4 describes the implementation of the proposed techniques on TinyOS, an operating system for networked sensors [Hill et al. 2000], as well as experimental evaluation in a network of one laptop and thirty MICAz motes. Section 5 discusses related work. Section 6 concludes this paper and points out some future research directions. The appendices give more information related to the proposed techniques.

2. ASSUMPTIONS

2.1 Assumptions of Sensor Networks

We assume the senders of authenticated broadcast messages are computationally powerful nodes (e.g., laptops), which also have sufficient power supply (e.g., charged in a vehicle). There are certainly scenarios where the senders of broadcast messages are regular, resource-constrained sensor nodes. Our techniques proposed in this paper do not apply to such cases.

We assume signature-based and/or μTESLA-based broadcast authentication can be used in sensor networks. This implies the following more specific assumptions. When signature-based broadcast authentication is used, we assume regular sensor nodes can perform a limited number of public key cryptographic operations, and can finish each operation in a reasonable amount of time. As discussed in the Introduction, this assumption has been validated in [Gura et al. 2004]. For example, based on the results in [Gura et al. 2004], a MICA2 mote can finish a 1024-bit RSA signature verification in about 0.43 seconds, and a 160-bit ECDSA signature verification in about 1.62 seconds. However, such public key cryptographic operations still consume substantially more resources (e.g., battery power) than symmetric cryptographic operations, and can be exploited by attackers to launch DoS attacks.

When signature-based broadcast authentication is used, we also assume that a packet transmitted in a sensor network is large enough to accommodate a public key signature. As discussed earlier, using IEEE 802.15.4, ZigBee-compliant sensor nodes (e.g., MICAz [Crossbow Technology Inc.]) can support packet payload up to 102 bytes [IEEE Computer Society 2003], despite the fact that the default payload size in TinyOS is only 29 bytes [Hill et al. 2000]. Such a packet can certainly include, for example, a 160-bit ECDSA signature, which requires 40 bytes. To confirm this assumption, we performed experiments with MICAz motes to measure the packet delivery rate at different distances when the packet payload size is 102 bytes. In our indoor experiments, the packet delivery rate for MICAz is over 90% when the distance is 100 feet, compared with close to 0% packet delivery rate for MICA2. Appendix B shows more details.

When μTESLA or its variation is used for broadcast authentication, we assume the clocks of all sensor nodes are loosely synchronized and maximum clock difference between any two nodes is known to all nodes. Moreover, we assume the sender can distribute the parameters required for μTESLA or its variation (e.g., key chain commitment, starting time) to all the receivers.

We do not assume any specific broadcast protocol. The broadcast protocol can be simply flooding, or probabilistic broadcast (e.g., [Ni et al. 1999; Stojmenovic et al. 2002; Levis et al. 2004]). However, we do assume that in order to propagate a broadcast packet to the entire network, it is necessary for *some* receivers to re-broadcast the packet. This is certainly true for all existing broadcast protocols for wireless sensor networks due to the

limited signal range.

2.2 Assumptions of Attackers

We assume the attacker can eavesdrop, inject, and modify packets transmitted in the network. We assume the attacker has access to computationally resourceful nodes such as laptops and workstations. In particular, the attacker may use multiple resource nodes in parallel to speed up his attacks. We assume the attacker may use multiple colluding nodes in different parts of the network (e.g., create wormholes between different parts of a network). We assume the attacker may compromise some nodes and learn the cryptographic secrets on them. However, the attacker cannot compromise the broadcast sender. Thus, the attacker cannot forge valid signatures when signature-based broadcast authentication is used, and nor can it forge valid MAC before the authentication key is disclosed in the case of μTESLA-based broadcast authentication.

Our goal in this paper is to develop lightweight techniques to mitigate the DoS attacks against broadcast authentication launched by such attackers. In other words, we would like to enable regular sensor nodes to quickly identify most forged packets (if not all) without performing the costly signature verifications or packet forwarding.

3. MESSAGE SPECIFIC PUZZLES

Our general approach is to use efficient cryptographic primitives to provide a weak authenticator along with broadcast authentication in each broadcast packet. When digital signatures are used for broadcast authentication, a sensor node does not have to verify the digital signature if the weak authenticator cannot be verified. Similarly, when μTESLA or its variation is used, a sensor node can discard broadcast packets (instead of forwarding them) when the weak authentication fails. In both cases, the weak authentication mechanism can mitigate the DoS attacks.

As discussed in the Introduction, the proposed weak authentication mechanism has some nice properties: The verification of a weak authenticator is very efficient, but forging a weak authenticator is time-consuming, though not infeasible. Moreover, it is computationally infeasible to forge a weak authenticator before the broadcast sender discloses some one-time (or short-lived) secret information. As a result, weak authenticators cannot be pre-computed. Even if an attacker has sufficient computational resources to forge one or more weak authenticators, it is difficult to reuse these forged weak authenticators. Thus, this weak authentication mechanism substantially increases the difficulty of launching successful DoS attacks against broadcast authentication.

We would like to emphasize that weak authenticators are not intended as a replacement of digital signatures or μTESLA-based approaches. Instead, they are used as an additional layer of protection to filter out forged broadcast packets so as to reduce the resource consumption (especially the energy consumption) due to DoS attacks.

For the sake of presentation, we refer to the data item used in each broadcast packet for authenticating the packet as the *broadcast authenticator*. It is a digital signature when signature-based broadcast authentication is used, and the combination of a MAC and a disclosed key when a μTESLA-based approach is used. Moreover, we refer to the data item for weak authentication as the *weak authenticator*.

In the following, we first present a strawman approach to illustrate the basic idea and the potential threats. We then gradually enhance this approach to obtain the final solution. For simplicity, we assume there is one broadcast sender and many receivers in the later

presentation. But our techniques can certainly be used when there are multiple broadcast senders.

3.1 Weak Authentication through One-Way Key Chains: A Strawman Approach

This strawman approach uses one-way key chains to provide weak authentication. One-way key chains have been used in several scenarios to provide efficient authentication. Examples include S/Key [Haller 1994], TESLA [Perrig et al. 2000], and its variations [Perrig et al. 2001; Liu and Ning 2003; 2004].

To generate a one-way key chain, the sender first selects a random value K_n as the last key in the key chain, and then repeatedly performs a (cryptographic) hash function, which is a one-way function, to compute all the other keys. That is, $K_i = F(K_{i+1})$, where F is the hash function and $0 \leq i \leq n-1$. With the hash function F, given K_j in the key chain, it is easy to compute all the previous keys K_i ($0 \leq i < j$), but it is computationally infeasible to compute any of the later keys K_i ($j < i \leq n$). Thus, with the knowledge of the initial key K_0, a receiver can authenticate any key in the key chain by merely performing hash function operations. The initial key K_0 is often called the *commitment* of the key chain. Figure 2 illustrates an example of one-way key chain.

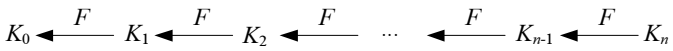

Fig. 2. An example one-way key chain. K_n is randomly generated. $K_i = F(K_{i+1})$, where F is a pseudo random function and $0 \leq i \leq n-1$. K_0 is used as the commitment of the key chain.

The sender can use these keys as weak authenticators. Before transmitting the broadcast packets, the sender distributes the commitment K_0 of the key chain to all the receivers. This can be done through, for example, pre-distribution. In this paper, we assume the commitment has been reliably distributed to all receivers. When the sender is ready to broadcast the i-th packet with message M_i, where $1 \leq i \leq n$, it first generates the broadcast authenticator BA_i (e.g., a digital signature). It then broadcasts the i-th packet, which includes the index i, the message M_i, the broadcast authenticator BA_i, and the i-th weak authenticator K_i.

Each receiver keeps the most recently authenticated weak authenticator K_j and the corresponding index j. Initially, $j = 0$ and $K_j = K_0$. Upon receiving a packet with index i, each receiver first checks the weak authenticator by verifying that (1) the i-th packet has not been been previously authenticated and (2) $K_j = F^{i-j}(K_i)$. The receiver discards the packet and stops if this verification fails. When signature-based broadcast authentication is used, it can further verify the broadcast authenticator BA_i (i.e., the signature). However, when μTESLA (or its variation) is used, the receiver cannot verify the broadcast authenticator immediately except for the disclosed key (for earlier packets) and the time-based security condition, until it receives the corresponding key disclosed in a later packet. Finally, the receiver replaces j with i, and K_j with K_i, and forwards the broadcast packet if necessary.

The use of one-way key chains provides some nice properties: Each weak authenticator K_i can be easily verified by regular sensor nodes. Moreover, before the broadcast of the i-th packet, an attacker does not have access to K_i, and thus cannot forge the weak authenticator (due to the one-way property of hash function F).

3.1.1 *Weakness of the Strawman Approach.* This strawman approach also has an obvious weakness. A malicious node may exploit an observed weak authenticator and the communication delay (or network partition) to forge broadcast packets, though it cannot forge a weak authenticator directly. Specifically, once a malicious node receives a broadcast packet, it may repeatedly replace the actual message and re-broadcast the modified packet. Moreover, it may use a fast channel (e.g., a wormhole [Hu et al. 2003]) to transmit the weak authenticator to another malicious node in a region that has not received the packet. The latter malicious node can then forge broadcast packets using this weak authenticator.

The strawman approach has a further implication for signature-based broadcast authentication. If the valid broadcast packet reaches the nodes being attacked within a reasonable amount of time, the number of signature verifications can still be bounded, since a receiver drops the packets whose index numbers belong to some previously verified packets. However, if the nodes being attacked are isolated from the sender due to network partition, the attacker can consume their battery power by forcing these nodes to perform an unbounded number of signature verifications. Note that μTESLA and its variations are not affected by such attacks, because broadcast packets are only valid for a limited period of time due to the time-based security condition (i.e., a broadcast packet is invalid when it is received after the corresponding authentication key may have been disclosed).

3.2 Message Specific Puzzles Based on One-Way Key Chains

In this subsection, we develop an initial version of message specific puzzles based on the strawman approach. We will improve it for signature-based and μTESLA-based broadcast authentication in the next subsections.

Our idea is to use cryptographic puzzles to reduce the possibility that an attacker may exploit an observed weak authenticator to forge broadcast packets. Intuitively, a sender (or an attacker) has to solve a cryptographic puzzle [Juels and Brainard 1999] in order to generate a valid weak authenticator. The puzzle solution is then used as the weak authenticator. A receiver can efficiently verify a weak authenticator; however, it takes an attacker a substantial amount of time to forge a weak authenticator.

Traditional cryptographic puzzles (e.g., client puzzles [Juels and Brainard 1999; Aura et al. 2001; Waters et al. 2004], congestion puzzles [Wang and Reiter 2004]) require interactions between a client and a server. However, broadcast in sensor networks, which involves one sender and a large number of receivers, does not permit such interactions. Moreover, we have to prevent an attacker from pre-computing puzzle solutions. Thus, we have to develop additional techniques to make this idea feasible.

Our solution is *keyed message specific puzzles based on one-way key chains* (or briefly, *message specific puzzles*). Intuitively, we consider each broadcast message, along with the message index and the broadcast authenticator, as a (message specific) puzzle. To prevent an attacker from pre-computing puzzle solutions to forged messages, we add in such a puzzle a previously undisclosed key in the one-way key chain. As a result, an attacker cannot pre-compute a puzzle solution until such a key is released by the sender. Upon receiving such a packet, any node can easily verify the puzzle solution. However, we develop the puzzle system in such a way that it will take a substantial amount of time to solve a puzzle. As a result, even if the key K_i is released in a broadcast packet, an attacker cannot immediately solve the puzzle for a forged packet, and thus cannot immediately launch DoS attacks.

3.2.1 *Basic Construction.* Now let us describe the details of message specific puzzles. As in the strawman approach, we assume the sender has generated a one-way key chain consisting of K_0, K_1, ..., K_n, and distributed K_0 to all potential receivers. The i-th key K_i ($1 \leq i \leq n$) in the one-way key chain is used for the weak authentication of the i-th broadcast packet. We also assume there is a hash function F_p known to the sender and all the receivers.

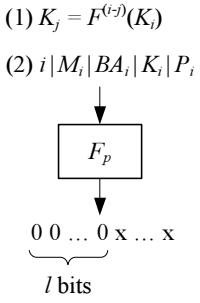

Fig. 3. Message specific puzzles

Given the i-th message M_i, the sender first generates the broadcast authenticator BA_i. The index i, the message M_i, the broadcast authenticator BA_i, and K_i then constitute the puzzle, which we call the *i-th message specific puzzle*. For the sake of presentation, we call K_i the *(i-th) puzzle key*, and denote the solution to this puzzle as P_i. As shown in Figure 3, a valid solution P_i to the i-th message specific puzzle, where $1 \leq i \leq n$, must satisfy the following two conditions:

(1) The puzzle key K_i is the i-th key in the one-way key chain, and

(2) After applying the hash function F_p to the i-th message specific puzzle and its solution, we get an image where the first l bits are all "0". That is,

$$F_p(i|M_i|BA_i|K_i|P_i) = \underbrace{00...0}_{l \text{ bits}}\text{xx...x},$$

where "xx...x" represents any bit pattern. The parameter l is called the *strength* of the puzzle.

Because of the one-way property of the hash function F_p, one has to search through the space of possible solutions to solve the puzzle. In other words, given i, M_i, BA_i, and K_i, for each candidate solution P_i', the sender (or an attacker) has to verify if the first l bits of $F_p(i|M_i|BA_i|K_i|P_i')$ are all "0". The sender is expected to try 2^l possible solutions before finding the right one, as we will show later in our analysis.

To take advantage of message specific puzzles, we use the puzzle key K_i (i.e., the i-th key in the one-way key chain) and the puzzle solution P_i together as the *weak authenticator for the i-th broadcast packet*. Given the i-th broadcast message M_i, the sender first generates the broadcast authenticator BA_i, retrieves the puzzle key K_i, and computes the puzzle solution P_i. The sender then broadcasts the packet with the payload $i|M_i|BA_i|K_i|P_i$.

Upon receiving a broadcast packet, each receiver first verifies the puzzle key using F and K_0 (or a previously verified puzzle key). Only when this verification is successful does the node verify the puzzle solution. If the puzzle solution is invalid, the receiver will simply drop this packet. Thus, without first solving some message specific puzzles, an attacker cannot force the nodes to verify digital signatures in forged packets (when signature-based broadcast authentication is used), nor can it force the nodes to forward forged packets (when μTESLA or its variation is used).

The requirement that the first l bits of the hash image are all "0" is an arbitrary decision. Indeed, the first l bits can be any fixed bit pattern. Another option is to have dynamic bit patterns that are changed periodically. However, this would require the synchronization of the sender and the receivers. Moreover, having dynamic bit patterns does not make

the puzzle solutions harder to find, since each puzzle solution is already protected with a (dynamic) puzzle key. Thus, we do not consider this option in our scheme.

Note that the use of puzzle keys is different from μTESLA. In μTESLA, the authentication key is disclosed after the sender is certain that all reachable receivers have received the corresponding broadcast packets. Thus, before the release of a μTESLA authentication key, the receivers cannot properly authenticate the received packets. This is the place where an attacker may exploit to launch DoS attacks. In contrast, message specific puzzles disclose a puzzle key in the same packet that uses this key. As a result, a receiver can immediately authenticate the puzzle solution.

Since the sender needs to solve a message specific puzzle before sending a broadcast packet, the computation involved in finding the puzzle solution should finish in a reasonable amount of time, though it should not be trivial to solve such a puzzle. Thus, an attacker may commit significant computational resources (e.g., multiple powerful computers) to compute puzzle solutions (and thus the weak authenticators) for forged packets once it obtains the puzzle key in a valid broadcast packet. When signature-based broadcast authentication is used, if the attacker is able to use a fast channel (e.g., a wormhole [Hu et al. 2003]) to send forged packets to nodes that have not received the valid broadcast packet, it may force these nodes to perform unnecessary signature verifications. When μTESLA-based approaches are used, the attacker may force the nodes to forward forged packets, which are indistinguishable from the valid one before the authentication key is disclosed. In both cases, the attacker can still consume sensor nodes' resources at the cost of solving puzzles.

3.2.2 *Minimizing Reuse of Forged Puzzle Solutions.* To mitigate the impact of forged packets, we have to reduce attacker's chances to reuse forged puzzle solutions. Otherwise, the attacker may compute only a few forged puzzle solutions, but force receivers to perform signature verifications or packet forwarding many times. In other words, we would like to ensure that an attacker has to pay more effort to generate higher impact.

We consider a puzzle solution in a received broadcast packet as a *forged* one if the puzzle solution is valid but the broadcast authenticator in the same packet is not. When signature-based broadcast authentication is used, a receiver can identify a forged puzzle solution after verifying the signature in the packet. However, when μTESLA-based approach is used, no receiver can detect forged puzzle solutions before the authentication key is disclosed. In this case, we consider each puzzle solution as a candidate of forged puzzle solution.

To minimize the impact of attacker reusing forged puzzle solutions, we may keep a buffer at each node for broadcast packets with potentially forged puzzle solutions. Specifically, we save the hash image of each broadcast packet in this buffer if the packet has a (potentially) forged puzzle solution. For brevity, we call this buffer the *packet hash buffer*. If an attacker reuses a previously forged packet, each receiver may identify the repeated transmission by searching in this buffer before verifying the digital signature or forwarding the broadcast packet.

Note that the above hash function does not have to have strong cryptographic properties (i.e., weak and strong collision free properties), given that a packet being hashed has to have a valid puzzle solution. The purpose of this hash is to derive a packet summary to identify a previously received packet. Such retransmitted packets will certainly result in the same hash images. However, given m hash images of previously received packets and a h-bit hash function, the probability of mistaking a fresh packet for one of the m previous

packets (i.e., rejecting an authentic broadcast packet as a forged one) is $\frac{m}{2^h}$. Thus, having, for example, a 40-bit hash function would provide sufficiently low probability of rejecting an authentic broadcast packet with a reasonable number of buffer entries (e.g., $m = 50$).

We use the *multi-buffer random selection* strategy in [Liu and Ning 2003; 2004] to manage the packet hash buffer. (Note that in signature-based broadcast authentication, this is no longer necessary for a given broadcast packet once the authentic one is received.) Specifically, assume each node has m entries in the packet hash buffer. For each incoming broadcast packet with a valid puzzle solution, each node first checks if the packet hash already exists in the buffer, and drops the packet if yes. Otherwise, for the packet with the k-th forged puzzle solution, if $k \leq m$, the node simply saves the packet hash in an empty buffer entry. If $k > m$, the node does not have enough buffer to save all forged packet hashes. In this case, the node saves it with probability $\frac{m}{k}$. If the packet hash is to be saved, the node randomly picks a buffer entry and replaces the old entry with the new one.

It is easy to see that when the attacker has more than m forged puzzle solutions, the more frequently the attacker uses one particular forged puzzle solution, the more possible the corresponding packet hash is in the buffer when it reaches a sensor node (and is then discarded). Thus, a good strategy for the attacker is to use these forged puzzle solutions at the same frequency. In this case, it is also easy to verify that given k' ($m < k' \leq k$) distinct forged puzzle solutions, each solution has the same probably $\frac{m}{k'}$ to have an entry in the buffer.

The sending procedure for both signature-based and μTESLA-based approaches are the same except for the generation of broadcast authenticators. However, the receiving procedure with which each node processes incoming broadcast packets varies slightly for signature-based and μTESLA-based broadcast authentication due to their difference in providing immediate authentication. In signature-based broadcast authentication, each receiver can verify the signature immediately after the weak authenticator. Thus, once the i-th broadcast packet is received and authenticated, each receiver can discard all the later packets claimed as the i-th packet. However, μTESLA-based broadcast authentication does not provide immediate authentication. As a result, all packets that pass weak authentication are potentially the "correct" packets, and have to be buffered and forwarded if necessary. In the latter case, it is critical to have an appropriate puzzle strength to reduce the number of packets forgeable by an attacker.

3.3 Analysis

In the following, we provide analysis for various aspects of the proposed scheme, including the cost of finding a puzzle solution, expected sender-side delay, choice of puzzle parameters, collision of valid puzzle solution and buffered forged puzzle solutions, security analysis, and performance overhead.

3.3.1 *Cost of Finding a Puzzle Solution.* Assume the hash function F_p is a pseudo random function. Given a puzzle strength l, the probability of finding a puzzle solution within x trials is $P_{x,l} = 1 - (1 - 2^{-l})^x$. Thus, the expected number of trials of finding a puzzle solution is

$$E\{x\} = \sum_{x=1}^{\infty} P_{x,l} \cdot x = \sum_{x=1}^{\infty} (1 - 2^{-l})^{x-1} \cdot 2^{-1} \cdot x = 2^{-l} \cdot \sum_{x=1}^{\infty} (1 - 2^{-l})^{x-1} \cdot x.$$

It is easy to compute that $\sum_{x=1}^{\infty} a^{x-1} \cdot x = \frac{1}{(1-a)^2}$ when $0 \leq a \leq 1$. Thus, we have $E\{x\} = 2^l$. In other words, on average it takes 2^l trials to find a solution to a puzzle with strength l. Figure 4 shows the relationship between $P_{x,l}$ and $\frac{x}{2^l}$ for several different values of l.

Figure 4 reveals an important issue: It may take more than 2^l trials to find a solution for a message specific puzzle with strength l. Indeed, there is no specific bound on the number of trials before a puzzle solution can be found. However, when the number of trials x grows large enough, a puzzle solution can be found with a very high probability. Assume the sender performs up to $x = 2^{l+c}$ trials, where c is a constant. In particular, consider $l = 128$, which represents a substantially strong puzzle. When $c = 6$, we can compute $P_{x=2^{128+6},l=128} = 1 - 1.6 \times 10^{-28}$. Moreover, we can prove that $P_{x=2^{l+c},l}$ decreases when l increases. (Detailed proof can be found in Appendix C.) This implies that when $l \leq 128$, the probability to find a solution to a puzzle with strength $l \leq 128$ within 2^{l+6} trials is at least $1 - 1.6 \times 10^{-28}$.

3.3.2 *Sender-Side Delay*. The variable cost of finding a puzzle solution has different implications in signature-based and μTESLA-based broadcast authentication. With signature-based broadcast authentication, the sender always uses the same private key to generate signatures. Given a broadcast message, the sender can simply sign the message, compute the puzzle solution, and broadcast the packet once a solution is found. The time used to generate a signature on a regular computer (e.g., around 20 ms in our experiments) is negligible compared with the time needed to solve a puzzle. Thus, the sender-side delay introduced by message specific puzzles is approximately proportional to the cost of finding a puzzle solution.

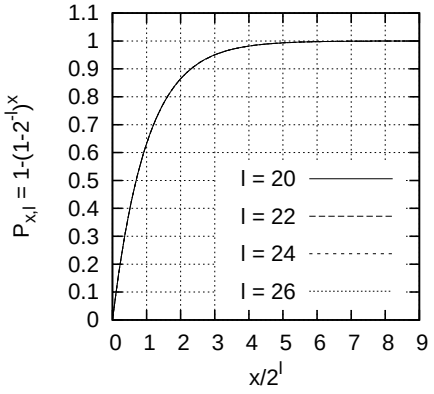

Fig. 4. Probability of finding a puzzle solution (Note that these lines almost overlap with each other)

However, μTESLA-based broadcast authentication uses different keys in different time intervals; the sender needs to use a key that will not be obsolete when the sender finds a puzzle solution. To address this issue, we propose to take a multi-round approach. When a broadcast message is to be sent, the sender first estimates the remaining time in the current μTESLA time interval. Because the cost of trying one puzzle solution is constant, the sender can then estimate the number X of possible puzzle solutions that can be tested before it is too late to authenticate it in the current time interval. (Note that we have to consider the time required to process and transmit the packet.) The sender can then determine the appropriate authentication key and generate the broadcast authenticator, assuming the puzzle can be solved in time. The sender then searches for up to X possible puzzle solutions. If a solution is found, the sender can then broadcast the packet. Otherwise, the sender can estimate the maximum number of puzzle solutions that can be tested for the next μTESLA time interval, and repeat the above process again with the next μTESLA key. (Due to the change in the μTESLA broadcast authenticator, the sender will have a different message specific puzzle.) This process may

continue until the right puzzle solution is found.

It is easy to see that the sender-side delay is still determined by the number of possible puzzle solutions the sender has tried before finding the right one. Thus, the sender-side delay for μTESLA-based broadcast authentication is approximately the same as that for signature-based broadcast authentication, though the sending procedure is different.

3.3.3 *Choice of Parameters.* We need to decide several parameters before we can use the message specific puzzles: the hash functions, the puzzle strength l, and the buffer size m for forged puzzle solutions.

Similar to the existing cryptographic puzzles (e.g., client puzzles [Juels and Brainard 1999; Aura et al. 2001; Waters et al. 2004], congestion puzzles [Wang and Reiter 2004]), we only use the one-way property (i.e., pre-image resistance) of cryptographic hash functions in message specific puzzles. Thus, as indicated in [Juels and Brainard 1999], we may use a fast hash function such as MD4 [Rivest 1992], or a fast block cipher such as RC6 [Rivest et al. 1998] as the hash function F_p.

Puzzle strength l is an important parameter for message specific puzzles. The decision of this parameter should follow two principles: First, the sender should be able to solve the puzzle within a reasonable amount of time. An overly large value for l will result in a long delay before transmitting broadcast packets on the sender's side. Second, the parameter should not be too small. In other words, the attacker should not be able to solve a large number of puzzles before the valid broadcast packet is propagated throughout the network. Based on these two principles, the network designer should determine the value l through balancing the maximum delay the sender can tolerate before sending the broadcast packet and the risk of DoS attacks against signature verifications.

The larger packet hash buffer a node has, the better it can minimize the reuse of forged puzzle solutions. In practice, parameter m should be determined based on the available storage on sensor nodes and the threat model. For example, when $l = 22$, we may use a 40-bit hash function to process potentially forged packets with valid puzzle solutions before saving the packet hashes. If there are more than 250 bytes available on each node, we may set $m = 50$. Based on the benchmark result for Crypto++ 5.2.1 [Dai 2004], it takes about 3.766 seconds on average for a 2.1 GHz Pentium 4 processor to solve one puzzle if SHA-1 is used. Thus, this setting can force an attacker with such a machine to spend about 196 seconds on average (after finding 52 solutions) in order to have a chance to reuse a previously forged puzzle solution.

3.3.4 *Security Analysis.* The one-way property of the hash function F_p brings a nice feature to message specific puzzles: An attacker has to search in a solution space in order to find a weak authenticator for a forged packet. As discussed earlier, given the puzzle strength l, an attacker needs to try 2^l hash function operations on average in order to find a puzzle solution. Moreover, the use of one-way key chains prevents an attacker from pre-computing puzzle solutions. In other words, it is computationally infeasible for an attacker to compute a puzzle key that has not been disclosed by the sender. Thus, the attacker cannot solve the message specific puzzle to forge the i-th broadcast packet until it has received a valid puzzle key K_i in the (real) i-th broadcast packet.

We temporarily assume that there is no network partition so that all broadcast packets can reach all the nodes in a finite amount of time. (We will discuss the case where there are network partitions later.) Due to the difficulty of solving message specific puzzles, given

an appropriate puzzle strength, an attacker may not have enough time to forge a weak authenticator before the broadcast packet reaches all sensor nodes if the attacker does not have substantial computational resources.

An attacker can certainly commit a lot of computational resources to forging weak authenticators. For each forged packet, the attacker has to solve a message specific puzzle, which involves on average 2^l hash function operations. Since each receiver has m entries in the packet hash buffer, the attacker cannot reuse any forged packet before he/she solves more than m puzzles. Consider the earlier example with puzzle strength $l = 22$ and 40-bit buffer entry. A 250 byte buffer for forged puzzle solutions will force an attacker with one 2.1 GHz Pentium 4 processor to compute for at least about 196 seconds on average before the attacker has a chance to reuse a forged puzzle solution.

Suppose the attacker has finished computing k' ($k' > m$) puzzle solutions, and is sending the k-th ($k > k'$) forged packet. In the best case, the attacker can succeed in reusing a previous puzzle solution with probability $1 - \frac{m}{k-1}$. This happens when the attacker sends a newly forged puzzle solution (i.e., the k'-th one) as the $(k-1)$-th packet and attempts to reuse it in the k-th packet. This probability will drop quickly as the attacker attempts to reuse the same forged puzzle solution.

Compared with the simple signature-based or μTESLA-based broadcast authentication, where an attacker can claim an arbitrary message as a broadcast packet and force many sensor nodes to verify signatures or forward packets, message specific puzzles have substantially increased the cost of DoS attacks. Moreover, as discussed earlier, even if the attacker has enough resources to launch such attacks, the forged weak authenticators are valid only for a limited period of time. In particular, when signature-based broadcast authentication is used, a forged broadcast packet has to arrive at a sensor node before the real packet to generate an impact.

Since each broadcast packet includes a message index for the sender, each message specific puzzle is unique. Moreover, the puzzle keys also change from packet to packet. Thus, puzzle solutions will also change with a high probability (approximately $1 - 2^{-l}$), and cannot be reused for later messages.

3.3.5 *Performance Overheads.* Message specific puzzles introduce light computational overhead on regular sensor nodes. For each broadcast packet, a receiver needs to perform a few hash function operations to verify the weak authenticator. When there are DoS attacks against signature verifications, the proposed approach can reduce the computational cost significantly by reducing the number of expensive signature verifications. However, the broadcast sender has to solve a message specific puzzle with strength l in order to generate a valid weak authenticator, which involves 2^l hash function operations on average per broadcast packet. Moreover, the sender needs to pre-compute a one-way key chain before the deployment of the network. This includes, for example, 10,240 hash function operations for a chain of 10,240 puzzle keys. As discussed earlier, we assume the broadcast sender is a powerful computer with external power supply, and can perform such operations.

Message specific puzzles require some space in each broadcast packet. Besides the message content and the digital signature, each packet has to include a message index, a puzzle key, and a puzzle solution. In general, a 16-bit index is enough for broadcast messages, and a 64-bit puzzle key is sufficient to prevent attacks against the one-way key chain. As discussed earlier, the solution to a message specific puzzle with strength l ($l \leq$

128) requires up to $l + 6$ bits space in the packet with at least probability $1 - 1.6 \times 10^{-28}$. They together require $2 + 8 + \lceil \frac{l+6}{8} \rceil$ bytes space in the packet (e.g., 14 bytes when $l = 24$). Considering the importance of broadcast authentication and the maximum payload size of 102 bytes in ZigBee-compliant sensor nodes (e.g., MICAz), such an overhead is acceptable. This overhead can be further reduced by 10 bytes for μTESLA, as we will show later.

The storage overhead on regular sensor nodes is reasonable. For both signature-based and μTESLA-based broadcast authentication, besides the data structure for signature or μTESLA, each node has to maintain the index of the most recently verified broadcast packet, the corresponding puzzle key, and the buffer for the hash images of m forged puzzle solutions. When 16-bit indexes, 64-bit puzzle keys, and a 40-bit hash function (for saving forged packets) are used, these require $10 + 5 \cdot m$ bytes space for each sender. For example, these require 260 bytes when $m = 50$. However, the storage requirement on the broadcast sender is much heavier. The sender has to keep at least the unused part of the one-way key chain, unless it computes the puzzle key every time it is needed. This requires, for example, 80,960 bytes for a chain of 10,240 64-bit keys. Given the assumption that the sender is a powerful node (e.g., a laptop), this is not a problem at all.

3.3.6 *A Remaining Threat.* A threat still remains for signature-based broadcast authentication when there are network partitions, even if we use message specific puzzles as weak authenticators. Consider the following scenario: A computationally resourceful attacker observes the i-th broadcast packet transmitted by the sender, and learns the puzzle key K_i. Thus, the attacker can forge the i-th broadcast packet with an invalid signature but valid weak authenticator. This is generally not a big threat to a connected network, because a node will discard the forged packets after receiving the valid broadcast packet. However, when some nodes are isolated from the sender (i.e., they cannot receive the packet from the sender), the attacker can repeatedly forge packets and send to these nodes, and thus force them to verify the (invalid) signatures.

It is easy to see that the above problem caused by network partition is not a threat to μTESLA-based approaches, because receivers can easily filter out forged packets when they do not satisfy the time-based security condition. That is, all nodes discard packets that are received after the corresponding keys are possibly disclosed.

3.4 Optimization for Signature-Based Broadcast Authentication

In the following we discuss two techniques to optimize the integration of message specific puzzles and signature-based broadcast authentication. We first enhance message specific puzzles to mitigate the aforementioned attacks against nodes isolated from the sender. For brevity, we call a node unreachable by the sender an *isolated node*.

3.4.1 *Time Limited Message Specific Puzzles.* The essential reason for the above attack is that a puzzle key remains valid for a node as long as this node has not authenticated a broadcast packet that uses this or a later key in the key chain. Our solution is thus to invalidate this condition.

Our solution is inspired by TESLA [Perrig et al. 2000]. As shown in Figure 5, we divide the time period for broadcasting into multiple time intervals, labeled as I_1, I_2, ..., I_n. Each puzzle key K_i in the one-way key chain is associated with the time interval I_i, where $1 \leq i \leq n$. The sender uses K_i for weak authentication only during the time interval I_i. For convenience, we denote the starting point and the end point of interval I_i ($1 \leq i \leq n$)

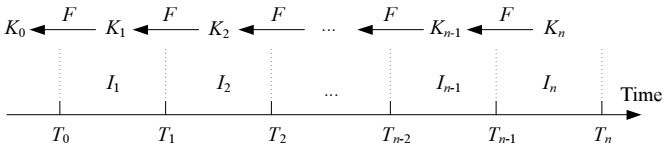

Fig. 5. One-way key chain in time-limited message specific puzzles. Each K_i is only valid between $T_{i-1} - \delta_c$ and $T_i + \delta_c + \delta_p$, where δ_c is the maximum clock difference between the sender and any receiver, and δ_p is the maximum propagation delay.

as T_{i-1} and T_i, respectively.

We assume the clocks of the sender and all receivers are loosely synchronized. More precisely, we assume the clock difference between any two nodes is bounded by δ_c. Moreover, we assume the propagation delay of any broadcast packet in a network without partition is bounded by δ_p. This delay may also include the time required by signature verification at intermediate forwarding nodes. At each receiver, each key K_i is only valid between $T_{i-1} - \delta_c$ and $T_i + \delta_c + \delta_p$ (in the local clock). When a node receives a broadcast packet at local time t with a weak authenticator, which is composed of the puzzle key K_i and the puzzle solution P_i, it first verifies the condition $T_{i-1} - \delta_c < t < T_i + \delta_c + \delta_p$, and continues to verify the packet only when this condition is satisfied. As a result, even if a node is isolated from the sender, an attacker can only use a cracked weak authenticator for a limited period of time.

Note that if the sender and the receivers are loosely synchronized, they can simply use TESLA or μTESLA instead of signature for broadcast authentication. However, when resources on the sensor nodes permit, it is desirable to use signatures (rather than TESLA or μTESLA) for broadcast authentication, because using signatures offers immediate authentication of received packets. Though the immediate authentication mechanism in [Perrig et al. 2001] provides receiver immediate authentication, it can be easily disrupted if there are packet losses. Therefore, using signatures along with message specific puzzles has some unique properties that cannot be offered by TESLA or μTESLA alone.

The reader may have noticed that when the sender has not broadcast for a relatively long period of time, all the receivers have to perform a potentially large number of hash function operations to verify the key in a new broadcast packet. A simple solution to this problem is to have the sender periodically (e.g., for every 100 time intervals) broadcast the most recently expired key to the network. (Note that such keys are self-authenticated because of the one-way key chain.) After receiving and authenticating such a key, each receiver replaces the most recently authenticated puzzle key with the new one. As a result, the receiving nodes can spread the verification of the puzzle keys in the one-way key chain over time.

Time limited message specific puzzles retain the security and performance properties of message specific puzzles discussed earlier. Moreover, it can prevent attackers from launching unlimited DoS attacks against isolated nodes, as discussed earlier. This extension does bring a restriction along with the benefit: The sender cannot send more than one broadcast packet per time interval, since each time interval has only one puzzle key. This can be addressed by having short time intervals, or having multiple puzzle keys per interval. The sender may need a potentially large number of puzzle keys, many of which are not used. Such a problem can be potentially addressed by using sandwich chains [Hu et al. 2005] or

multi-level key chains [Liu and Ning 2003; 2004]. These approaches provide more complex but efficient ways to organize key chains, and allow receivers to skip the computation of intermediate keys when authenticating later keys. Since these are not the focus of this paper, we do not discuss them in detail.

3.4.2 *Adaptive Verification.* As discussed earlier, broadcast in a wireless sensor network typically requires that some nodes receiving an authenticated broadcast packet re-broadcast it (locally) to propagate the packet across the network. In the proposed (time limited) message specific puzzles, such a node verifies the puzzle solution and the digital signature before forwarding the broadcast packet. Though the verification of solutions to message specific puzzles is trivial, signature verification takes much longer time. This will certainly introduce undesirable delays in large sensor networks.

An alternative approach is to have each node re-broadcast the packet right after verifying the puzzle solution but before verifying the signature. However, message specific puzzles are *weak* authenticators intended for mitigating DoS attacks against broadcast authentication. As discussed earlier, they can be forged if the attacker devotes significant computational resources. If a node uses this alternative approach, it may forward forged packets before realizing that they are forged.

It seems that both approaches are not satisfactory. To address this dilemma, we propose an adaptive approach to determining the order of signature verification and forwarding of broadcast packets. Intuitively, this approach tries to detect attempts of DoS attacks against signature verifications. In normal situations where there are no such attacks, each node re-broadcasts a broadcast packet once the weak authenticator is verified, and then verifies the signature. However, when there are DoS attacks against signature verifications, each node first verifies the digital signature, and then re-broadcasts the packet if the signature is valid.

Figure 6 illustrates this approach. Each node works in two modes: *optimistic mode* and *pessimistic mode*. In the optimistic mode, a node re-broadcasts the packet locally once it verifies the weak authenticator. In contrast, in the pessimistic mode, a node verifies both the weak authenticator and the signature, and re-broadcasts the packet only when both verifications pass. The switch between these two modes is determined by a detection metric N_f, *the number of failed signature verifications in the past* w *time units*, where w is a system parameter determined by the security policy. Note that a node verifies a signature only when the weak authenticator is valid. Thus, N_f represents the number of forged broadcast packets with valid weak authenticators but invalid signatures. A node initially works in the optimistic mode. It switches to pessimistic mode if N_f becomes greater than 0, and may switch back to the optimistic mode when N_f drops to 0.

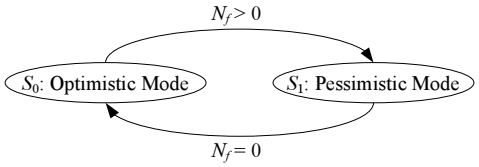

Fig. 6. Adaptive verification (N_f: # of failed signature verifications in the past w time units)

Adaptive verification can be used with either message specific puzzles or time limited message specific puzzles, and retains the same security properties. When there are DoS attacks, this approach is exactly the same as proposed earlier. However, in normal situ-

ations where there are no such attacks, adaptive verification can substantially reduce the broadcast delay.

3.5 Optimization for μTESLA-Based Broadcast Authentication

As discussed earlier, both message specific puzzles and μTESLA use one-way key chains. In this subsection, we propose to reuse the μTESLA key chain for message specific puzzles. An immediate benefit is that only one key needs to be disclosed in a broadcast packet. Consider the small packet size typically allowed in low-power wireless communication (e.g., 102 bytes payload size in ZigBee standards). Such an approach provides space savings that could be important for the applications. For example, suppose we use message specific puzzles with strength $l = 22$ and reserve 4 bytes for a puzzle solution. Further assume we use 64-bit keys. This approach reduces the space overhead introduced by message specific puzzles from 14 bytes per packet to 4 bytes per packet (due to the reuse of the index and the disclosed key), resulting in a 71% reduction in space requirement. Moreover, by using the same key chain for both μTESLA and message specific puzzles, the management of key chains becomes easier.

Note that we cannot use a μTESLA key that has not been disclosed as a puzzle key, since the puzzle key has to be released in a broadcast packet. Moreover, we should try to avoid using a key disclosed earlier, because an attacker may use such a key to forge weak authenticators once he/she learns the key. Thus, the best choice is to use the most recently disclosed μTESLA key as the puzzle key. Specifically, we propose to reuse the μTESLA key disclosed in each packet as the puzzle key for this packet. Except for the choice of puzzle keys, this approach works exactly the same as the basic construction presented in Section 3.2.

It may appear that this approach also introduces some drawbacks due to the repeated use of the same puzzle key. In other words, since the same μTESLA key may be disclosed in all packets broadcast in the same time interval, it is a valid puzzle key for an entire time interval, and may be re-used as the puzzle key for multiple packets. An attacker may exploit this fact to forge more weak authenticators once he/she learns the puzzle key.

However, we show this is indeed not the case. Consider the basic message specific puzzle construction for μTESLA-based broadcast authentication. It is known that μTESLA-based approaches do not provide immediate authentication. As a result, a receiver cannot fully authenticate a broadcast packet until the corresponding μTESLA key is disclosed. Thus, an attacker can use a puzzle key learned from a broadcast packet to forge as many weakly authenticated packets (i.e., packets with valid weak authenticators) as permitted by his/her computational resources. However, a receiver can still partially detect forged packets using the disclosed μTESLA keys. Specifically, if a weakly authenticated packet does not have the most recently disclosed μTESLA key, it can be identified as a forged packet. Moreover, if a puzzle key has been used in a previous time interval, the packets weakly authenticated in a later time interval using the same puzzle key must all be forged. This is because in the basic message specific puzzle construction, puzzle keys are not reused. Thus, in the basic message specific puzzle construction for μTESLA-based broadcast authentication, the valid period during which an attacker can reuse a puzzle key to forge packets is about one time interval in μTESLA. This is exactly the same as the proposed optimization in this subsection.

3.6 Limitations

Despite the useful properties, message specific puzzles also have some limitations. First, the broadcast sender has to solve a puzzle before broadcasting a message. This requires that the sender must be a computationally powerful node with sufficient power supply, and also implies that there will be a delay before the transmission of the packet. However, in certain applications (e.g., task dissemination without real-time requirements), these problems are tolerable in exchange of the mitigation of the DoS attacks.

One may be concerned that the sender-side delay may accumulate when the sender needs to transmit a large number of packets to broadcast a large amount of data in a short period of time. Because every broadcast packet may require a message specific puzzle and introduce a delay, the aggregated delay could be substantial. A good example is network based reprogramming, during which the sender needs to propagate a new program image to all the sensor nodes. Fortunately, in such cases, we do not have to digitally sign (or authenticate with μTESLA) every single packet. For example, Deng et al. [2006] gives an approach to only sign the first packet, which authenticates the hash images of the later packets. As a result, only the first packet for the entire program image needs the protection of message specific puzzles. Such techniques can certainly be used for other bulk data broadcast besides remote network reprogramming.

Besides the requirement of resourceful senders and the sender-side delays, message specific puzzles add moderate communication overhead and storage overhead on regular sensor nodes. As discussed earlier, these overheads are generally acceptable on the recent sensor network platforms such as MICAz and TelosB, especially when they are not frequently used.

4. IMPLEMENTATION AND EXPERIMENTAL EVALUATION

We have implemented the proposed techniques on TinyOS [Hill et al. 2000], and performed initial experimental evaluation. Our goal here is to understand performance issues that cannot be obtained directly through theoretical analysis. In the following, we first describe our implementation and then present the evaluation results.

4.1 Implementation

We use TinyECC [Liu and Ning], a software package for Elliptic Curve Cryptography (ECC) on TinyOS, for signature-based broadcast authentication. We implemented μTESLA on TinyOS for μTESLA-based broadcast authentication, in which we used RC5 CBC-MAC in TinyOS as the message authentication code. To focus on the performance issues, we pre-distribute all the key chain commitments and the sender's public key (in case of signature-based broadcast authentication), and synchronize the sensors' clocks before the experiments. To allow the transmission of broadcast packets with ECDSA signatures, we revised the maximum payload size in TinyOS from 29 bytes to 102 bytes, which is the maximum payload size in IEEE 802.15.4 standard specification [IEEE Computer Society 2003].

We reuse the SHA-1 implementation in the TinyECC package as the hash functions for both message specific puzzles and one-way key chains. To reduce the size of the puzzle keys included in broadcast packets, when generating the one-way key chain, we randomly generate a 64-bit key as the last puzzle key (K_n), and truncate the output of SHA-1 function to 64 bits. Thus, all puzzle keys in a one-way key chain have 64 bits. Note that truncating

each SHA-1 output to 64 bits does not necessarily provide the expected security as in a 64-bit one-way function. This is simply an implementation decision, and can be replaced if necessary.

We refer to our implementation of the message specific puzzle mechanisms as *TinyBcastGuard*. Due to the slight differences and optimizations that message specific puzzles have for signature-based and μTESLA-based broadcast authentication, we implemented them in two separate software packages: *TinySigGuard* and *TinyμTESLAGuard*. TinySigGuard implements the message specific puzzles for signature-based broadcast authentication; it consists of two parts: *TSGSender* and *TSGReceiver*. Similarly, *TinyμTESLAGuard* implements the message specific puzzles for μTESLA-based broadcast authentication, and also consists of two parts: *TuGSender* and *TuGReceiver*.

TSGSender and TuGSender are Java programs running on a PC. They communicates with the sensor network through a regular sensor node attached to the PC, which runs *TOSBase*, an application (in the TinyOS distribution) that simply forwards packets between the sensor network and the PC. To broadcast an authenticated packet, both of them generate the packet by first creating the broadcast authenticator based on the broadcast data and solving the message specific puzzle. They then send the packet to the node running TOSBase to broadcast the packet.

TSGReceiver and TuGReceiver run on regular sensor nodes, and are responsible for verifying the weak authenticators and broadcast authenticators (i.e., digital signatures, μTESLA disclosed keys and MACs) in broadcast packets and re-broadcasting the packets. We take the simplest flooding approach as the broadcast protocol. That is, each receiver re-broadcasts a packet once the packet is authenticated or weakly authenticated, depending on the approach. It is certainly desirable to experiment with other more efficient broadcast protocols; we will do so in our future research. Table I shows the code sizes of TSGReceiver and TuGReceiver on MICAz, obtained using the `check_size.pl` script in the TinyOS CVS repository. The code size of TinyECC and μTESLA are not included.

Table I. Code size (bytes) on MICAz

	ROM	RAM
TSGReceiver	1,317	289
TuGReceiver	180	210

Figure 7 gives the packet formats for signature-based and μTESLA-based broadcast authentication. In case of signature-based broadcast authentication, i is the packet index, M_i is the broadcast message, Sig is an ECDSA signature, K_i is the puzzle key, and P_i is the puzzle solution. In case of μTESLA-based broadcast authentication, the packet further has j, the index of the disclosed μTESLA key, and a MAC (authenticated with K_{j+d}) is used instead of an ECDSA signature.

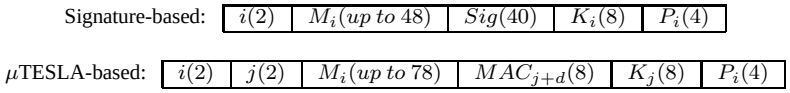

Fig. 7. Broadcast packet format (bytes)

4.2 Experimental Evaluation

4.2.1 *Experiment Scenario.* We evaluated our implementation in a testbed consisting of one laptop sender (connected to a MICAz mote through a programming board) and thirty regular sensor node receivers. The sender is a DELL Latitude D510 laptop with a

1.6 GHz Pentium M 730 processor and 512 MB DDR SDRAM. Each sensor node is a MICAz mote, which has an 8-bit Atmega128 processor and an IEEE 802.15.4 compliant RF transceiver. (More details about MICAz can be found in [MIC].) As mentioned earlier, our implementation uses Java for the senders. To better understand the timing results in practice, we also used Crypto++ Library 5.2.1 (`http://www.eskimo.com/~weidai/cryptlib.html`) in some experiments to obtain the execution time.

Figure 8 shows the experiment scenario. The laptop sender communicates with the sensor network through the MICAz mote on the programming board, which runs *TOSBase*. The sender periodically broadcasts a 10-byte broadcast message. When the sender needs to broadcast a message, it generates a broadcast packet by first creating the broadcast authenticator and then solving the message specific puzzle. It then sends the packet to the node running TOSBase to broadcast the packet. The receivers are responsible for verifying the weak authenticator and broadcast authenticator (i.e., digital signatures, μTESLA disclosed keys and MACs) in each broadcast packet and forwarding the packet. We take the simplest flooding approach as the broadcast protocol. That is, each receiver re-broadcasts a packet once the packet is authenticated or weakly authenticated, depending on which broadcast authentication approach is used. It is certainly desirable to experiment with other more efficient broadcast protocols; we will do so in our future research.

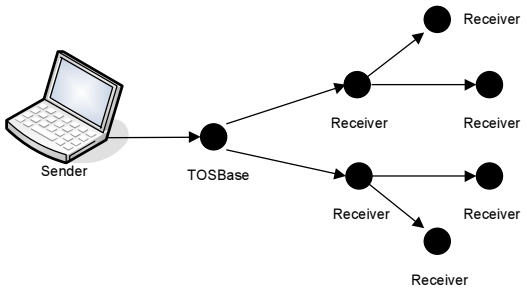

Fig. 8. The experiment scenario

4.2.2 *Computational Cost and Sender-Side Delay.* We measure the execution time required at the sender and each receiver for the generation and verification of message specific puzzle solutions. The time required at the sender directly affects the sender-side delay.

Let us first consider signature-based broadcast authentication with message specific puzzles. Table II shows the expected time required to solve a message specific puzzle for signature-based broadcast authentication (on the PC using Java and Crypto++, respectively) and verifying a puzzle solution (on a MICAz mote). It is easy to see that verifying a puzzle solution at a receiver is extremely efficient. The sender-side delay is simply the time required to generate the broadcast authenticator plus the time required to search for a puzzle solution. Though signature generation is generally more expensive than μTESLA MAC generation, it is still very efficient compared with solving a puzzle. For example, in the aforementioned platform, the ECDSA implementation provided in J2SE 5.0 and Bouncy Castle JCE provider (`http://www.bouncycastle.org`) can generate a 160-bit signature in about 30 ms. It is easy to see the delay introduced by signature generation is

Table II. Expected time (millisecond) required for solving and verifying a message specific puzzle

Puzzle Strength (l)	signature-based		Verifying a Solution (MICAz)
	Java	Crypto++	
20	8,357	2,590	14.6
22	35,220	10,377	14.6
24	142,237	41,440	14.6
26	599,953	165,893	14.6

Table III. Average sender-side delay (millisecond) for μTESLA-based broadcast authentication

Puzzle Strength (l)	Duration of each μTESLA interval			
	500 ms	1000 ms	2000 ms	4000 ms
20	7,976	7,337	7,265	6,033
22	26,115	26,073	25,929	25,390
24	105,231	105,219	104,922	104,842
26	431,359	431,031	430,438	429,688

negligible. Thus, the sender-side delay is approximately the time required for solving message specific puzzles when signature-based broadcast authentication is used.

When μTESLA-based broadcast authentication with message specific puzzles is used, the cost of verifying a puzzle solution at a mote remains the same as in Table II. However, the computational cost at the sender and the expected sender-side delay are more complicated, because the sender may have to generate MACs again when different μTESLA keys are used. We measured the aggregated computational cost experimentally using the aforementioned laptop. Table III shows the sender-side delays obtained using TuGSender with different puzzle strengths and μTESLA time intervals. (Note that TuGSender is written in Java.) When puzzle strength is fixed, the average sender-side delay decreases as μTESLA time interval increases. This is because the overhead due to the switches to different μTESLA keys will decrease when the duration of each time interval increases. The cost for μTESLA-based broadcast authentication is in general smaller than that for signature-based approach in Table II, because the space overhead introduced by μTESLA (10 bytes) is smaller than that by 160-bit ECDSA signatures (40 bytes), leading to smaller input of the message specific puzzle.

The puzzle strength offers a tradeoff between the sender-side delay and the resilience against DoS attacks. For critical applications that require high resilience against potential DoS attacks launched by highly resourceful attackers, it is reasonable to use a high puzzle strength. As a result, even if the attacking node has high computational resource, it cannot force regular sensor nodes to perform a large number of unnecessary signature verifications or message transmissions, despite the resulting long sender-side delay. However, it is not always desirable to use a puzzle strength that introduces a long sender-side delay. In non-critical applications, it usually does not justify the high cost for the attackers to deploy resourceful attacking nodes such as a laptop computer. Thus, we may choose to use short puzzle strengths, which can effectively defeat DoS attacks launched by regular sensor nodes (e.g., MICAz motes) without introducing long sender-side delays.

4.2.3 *Propagation Delay.* We have performed experiments to measure the propagation delay for signature-based and μTESLA-based broadcast authentication with and without weak authentication in the testbed. We used ECDSA on the 160-bit elliptic curve

`secp160k1` specified by SECG [Certicom Research 2000]. As discussed earlier, we used a simple flooding protocol for broadcast. That is, each node re-broadcasts an authenticated packet when it receives this packet for the first time. To reduce packet collisions, each node randomly delays between 0 and 50 milliseconds before re-broadcasting.

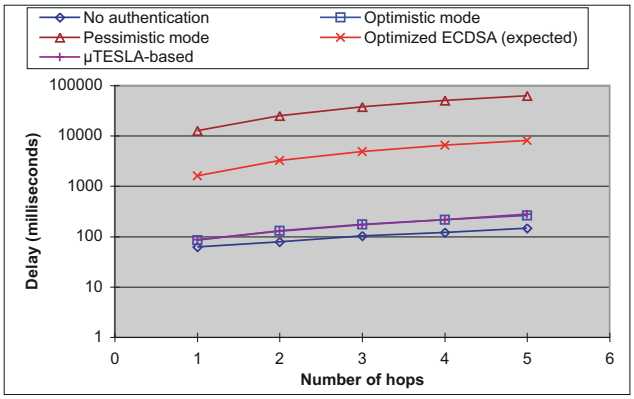

Fig. 9. Propagation delays

Figure 9 shows the broadcast delays at different hops from the sender for five cases: (1) no broadcast authentication, (2) signature-based broadcast authentication in optimistic mode, (3) signature-based broadcast authentication in pessimistic mode, (4) signature-based broadcast authentication in pessimistic mode using the optimized ECC implementation in [Gura et al. 2004], and (5) μTESLA-based approach. The first three cases and the fifth case were obtained in our experiments, while the fourth case is estimated based on the timing results in [Gura et al. 2004] (i.e., each ECDSA signature verification takes about 1.62 seconds).

It is easy to see that the signature-based broadcast authentication with message specific puzzles introduces light delays when used in optimistic mode. However, in pessimistic mode (i.e., when there are DoS attacks), this approach does add significant delays (e.g., about 8 seconds to reach 5 hops with the optimized ECC implementation in [Gura et al. 2004]). Though these results do not justify the immediate use of these techniques, they are close to acceptable performances. We expect these techniques will be practical when sensor nodes with better processing power are available.

We can see from Figure 9 that the propagation delay for μTESLA-based broadcast authentication with message specific puzzles is very close to that for signature-based broadcast authentication in optimistic mode due to the light computation at the receivers. This implies that message specific puzzles for μTESLA-based broadcast authentication can be used efficiently for networks of MICAz motes.

We have performed initial experimental evaluation in normal situations. It is also desirable to experiment with these techniques when there are attacks. We will perform such experiments in our future research.

5. RELATED WORK

Broadcast authentication has been traditionally achieved with digital signatures, where the sender signs the messages and all the receivers can authenticate the messages by verifying

the signatures. In the past few years, many researchers have been working on how to reduce the number of signature operations in, for example, streaming applications over lossy channels (e.g., graph-based broadcast authentication [Gennaro and Rohatgi 1997; Song et al. 2002; Miner and Staddon 2001], forward error correction based approaches [Park et al. 2003; Pannetrat and Molva 2003]). In addition, DoS protection in stream broadcast authentication have been investigated [Karlof et al. 2004; Gunter et al. 2004]. Gunter et al. [2004] proposed a *selective verification* technique to reduce the number of unnecessary signature verifications and examinations of packet hashes, while Karlof et al. [2004] developed distillation code to limit the number of packet combinations that have to be verified together. In contrast, the DoS attacks in our paper is about the verification of individual packets, and the proposed message specific puzzle technique is to reduce the cost involved in the verification of individual packets. The message specific puzzles are complementary with the previous DoS protection techniques.

Researchers have been working on broadcast authentication purely based on symmetric cryptography, such as TESLA [Perrig et al. 2000] and its variations [Perrig et al. 2001; Liu and Ning 2003; 2004; Liu et al. 2005], BiBa [Perrig 2001], and HORS [Reyzin and Reyzin 2002]. In particular, μTESLA [Perrig et al. 2001] and the later variations have been considered a good candidate for broadcast authentication in wireless sensor networks. As discussed earlier, a major limitation of μTESLA and its variations is the lack of immediate authentication, which could be exploited by an attacker to launch DoS attacks. The techniques developed in this paper target at mitigating such DoS attacks against μTESLA-based (and signature-based) broadcast authentication, aiming at practical broadcast authentication in sensor network applications.

Message specific puzzles are essentially an integration of client puzzles and one-way hash chains. Client puzzles were proposed in [Juels and Brainard 1999] and later improved in several application contexts, including defense of DoS attacks against secure web servers [Dean and Stubblefield 2001], DoS-resistant authentication protocols [Aura et al. 2001], distributed puzzles for mitigating bandwidth-exhaustion attacks [Wang and Reiter 2004], and delegated distribution of puzzles [Waters et al. 2004]. However, all the previous cryptographic puzzle techniques require interactions between a client and a server. Another technique closely related to the proposed approach as well as client puzzles is Hashcash [Back 2002], which uses the finding of partial hash collisions as a proof of work. Hashcash has a non-interactive version; however, it allows pre-computation attacks, and thus cannot be used for our purposes. Dwork and Naor [1992] proposed to use client puzzles with shortcuts to control resource usage, particularly the generation of junk mails. Similar to non-interactive hashcash, this approach allows pre-computation and is not suitable for our purposes. Our innovation in this paper is to integrate cryptographic puzzles, one-way key chains, and broadcast messages together to achieve weak authentication without requiring interaction between the sender and many receivers.

Our research is also related to DoS attacks and defenses in wireless sensor networks. In particular, Xu et al. [2005] studied the feasibility of launching and detecting jamming attacks in wireless networks. Cagalj et al. [2006] proposed to exploit channel diversity in order to create wormholes to defend against physical jamming attacks. Our techniques proposed in this paper are complementary to these researches in defending against DoS attacks.

6. CONCLUSION

In this paper, we developed message specific puzzles, a weak authentication mechanism, to mitigate DoS attacks against signature-based and μTESLA-based broadcast authentication in wireless sensor networks. This approach has a number of nice properties: First, a weak authenticator can be efficiently verified by a regular sensor node, but takes a computationally powerful attacker a substantial amount of time to forge. Second, a weak authenticator cannot be pre-computed without a non-reusable (or short-lived) key disclosed only in a valid broadcast packet. Thus, an attacker cannot start the expensive computation to forge a weak authenticator without seeing a valid broadcast packet. Third, even if an attacker has sufficient computational resources to forge one or more weak authenticators, it is difficult to reuse these forged weak authenticators. Thus, this weak authentication mechanism substantially increases the difficulty of launching successful DoS attacks against signature-based and μTESLA-based broadcast authentication. A limitation of this approach is that it requires a powerful sender and introduces sender-side delay due to the computation of puzzle solutions. We have implemented the proposed techniques in on TinyOS, and performed initial experimental evaluation in a network of MICAz motes.

In our future research, we will seek solutions that can provide weak authentication without requiring significant computational power at the sender. Moreover, we will continue the experimental evaluation in large-scale sensor networks, and investigate the integration with efficient broadcast protocols for wireless sensor networks.

7. ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments.

REFERENCES

Micaz: Wireless measurement system. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.

2006. 2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver. http://www.chipcon.com/files/CC2420_Data_Sheet_1_4.pdf.

AKYILDIZ, I., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. 2002. Wireless sensor networks: A survey. *Computer Networks 38,* 4, 393–422.

AURA, T., NIKANDER, P., AND LEIWO, J. 2001. DOS-resistant authentication with client puzzles. In *Proceedings of the 8th International Workshop on Security Protocols, LNCS 2133*. 170–177.

BACK, A. 2002. Hashcash – a denial of service counter-measure. http://www.cypherspace.org/hashcash/hashcash.pdf.

CAGALJ, M., CAPKUN, S., AND HUBAUX, J.-P. 2006. Wormhole-based anti-jamming techniques in sensor networks (to appear). *IEEE Transactions on Mobile Computing*.

CERTICOM RESEARCH. 2000. Standards for efficient cryptography – SEC 2: Recommended elliptic curve domain parameters. http://www.secg.org/collateral/sec2_final.pdf.

CROSSBOW TECHNOLOGY INC. Wireless sensor networks. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

DAI, W. 2004. Crypto++ 5.2.1 benchmarks. http://www.eskimo.com/~weidai/benchmarks.html.

DEAN, D. AND STUBBLEFIELD, A. 2001. Using client puzzles to protect TLS. In *Proceedings of the 10th USENIX Security Symposium*.

DENG, J., HAN, R., AND MISHRA, S. 2006. Secure code distribution in dynamically programmable wireless sensor networks. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN '06)*.

DWORK, C. AND NAOR, M. 1992. Pricing via processing, or, combatting junk mail. In *Advances in Cryptology – CRYPTO 92*. Lecture Notes in Computer Science, vol. 740. 139–147.

GAY, D., LEVIS, P., VON BEHREN, R., WELSH, M., BREWER, E., AND CULLER, D. 2003. The nesC language: A holistic approach to networked embedded systems. In *Proceedings of Programming Language Design and Implementation (PLDI 2003)*.

GENNARO, R. AND ROHATGI, P. 1997. How to sign digital streams. In *Advances in Cryptology – CRYPTO '97*. 180–197.

GUNTER, C., KHANNA, S., TAN, K., AND VENKATESH, S. 2004. DoS protection for reliably authenticated broadcast. In *Proceedings of the 11th Network and Distributed Systems Security Symposium (NDSS '04)*. 17–36.

GURA, N., PATEL, A., AND WANDER, A. 2004. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004)*. 119–132.

HALLER, N. M. 1994. The S/KEY one-time password system. In *Proceedings of the ISOC Symposium on Network and Distributed System Security*. 151–157.

HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. S. J. 2000. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*. 93–104.

HU, Y., JAKOBSSON, M., AND PERRIG, A. 2005. Efficient constructions for one-way hash chains. In *Proceedings of the 3rd International Conference on Applied Cryptography and Network Security*. 423–441.

HU, Y., PERRIG, A., AND JOHNSON, D. 2003. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings of INFOCOM 2003*.

IEEE COMPUTER SOCIETY. 2003. IEEE 802.15.4: Ieee standard for information technology – telecommunications and information exchange between systems local and metropolitan area networks – specific requirements part 15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs). `http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf`.

INTEL RESEARCH. Intel mote. `http://www.intel.com/research/exploratory/motes.htm`.

JUELS, A. AND BRAINARD, J. 1999. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *Proceedings of the 6th Network and Distributed Systems Security Symposium (NDSS '99)*.

KARLOF, C., SASTRY, N., LI, Y., PERRIG, A., AND TYGAR, J. 2004. Distillation codes and applications to dos resistant multicast authentication. In *Proceedings of the 11th Network and Distributed Systems Security Symposium (NDSS '04)*. 37–56.

LEVIS, P., PATEL, N., CULLER, D., AND SHENKER, S. 2004. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the 1st Symposium on Network System Design and Implementation (NSDI '04)*.

LIU, A. AND NING, P. Tinyecc: Elliptic curve cryptography for sensor networks (version 0.1). `http://discovery.csc.ncsu.edu/software/TinyECC/`.

LIU, D. AND NING, P. 2003. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS'03)*. 263–276.

LIU, D. AND NING, P. 2004. Multi-level μTESLA: Broadcast authentication for distributed sensor networks. *ACM Transactions in Embedded Computing Systems (TECS) 3*, 4, 800–836.

LIU, D., NING, P., ZHU, S., AND JAJODIA, S. 2005. Practical broadcast authentication in sensor networks. In *Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2005)*.

MINER, S. AND STADDON, J. 2001. Graph-based authentication of digital streams. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*. 232–246.

NEWSOME, J. AND SONG, D. 2003. GEM: graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys '03)*. 76–88.

NI, S., TSENG, Y., CHEN, Y., AND SHEU, J. 1999. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99)*. 151–162.

NICULESCU, D. AND NATH, B. 2001. Ad hoc positioning system (APS). In *Proceedings of IEEE GLOBECOM '01*.

PANNETRAT, A. AND MOLVA, R. 2003. Efficient multicast packet authentication. In *Proceedings of the 10th Network and Distributed Systems Security Symposium (NDSS '03)*. 251–262.

PARK, J., CHONG, E., AND SIEGEL, H. 2003. Efficient multicast stream authentication using erasure codes. *ACM Transactions on Information and System Security 6*, 2, 258–285.

PERRIG, A. 2001. The BiBa one-time signature and broadcast authentication protocol. In *Proceedings of the ACM Conference on Computer and Communications Security*. 28–37.

PERRIG, A., CANETTI, R., SONG, D., AND TYGAR, D. 2000. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*.

PERRIG, A., CANETTI, R., SONG, D., AND TYGAR, D. 2001. Efficient and secure source authentication for multicast. In *Proceedings of Network and Distributed System Security Symposium*.

PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D., AND TYGAR, D. 2001. SPINS: Security protocols for sensor networks. In *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks*. 521–534.

REYZIN, L. AND REYZIN, N. 2002. Better than BiBa: Short one-time signatures with fast signing and verifying. In *The 7th Australasian Conference on Information Security and Privacy*.

RIVEST, R. 1992. The MD4 message-digest algorithm. RFC 1320. `http://www.ietf.org/rfc/rfc1320.txt`.

RIVEST, R., ROBSHAW, M., SIDNEY, R., AND YIN, Y. 1998. The RC6 block cipher. Presented at the NIST Fist AES Candidate Conference.

SONG, D., ZUCKERMAN, D., AND TYGAR, J. 2002. Expander graphs for digital stream authentication and robust overlay networks. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*.

STOJMENOVIC, I., SEDDIGH, M., AND ZUNIC, J. 2002. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on Parallel and Distributed Systems 13*, 1, 14–25.

WANG, X. AND REITER, M. 2004. Mitigating bandwidth-exhaustion attacks using congestion puzzles. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS '04)*. 257–267.

WATERS, B., JUELS, A., HALDERMAN, J., AND FELTEN, E. 2004. New client puzzle outsourcing techniques for dos resistance. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS '04)*. 246–256.

XU, W., TRAPPE, W., ZHANG, Y., AND WOOD, T. 2005. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc 05)*. 46–57.

A. A BRIEF OVERVIEW OF μTESLA

An asymmetric mechanism such as public key cryptography is generally required for broadcast authentication [Perrig et al. 2000]. Otherwise, a malicious receiver can easily forge any packet from the sender, as discussed earlier. μTESLA introduces asymmetry by delaying the disclosure of symmetric keys [Perrig et al. 2001]. A sender broadcasts a message with a Message Authentication Code (MAC) generated with a secret key K, which is disclosed after a certain period of time. When a receiver gets this message, if it can ensure that the packet was sent before the key was disclosed, the receiver buffers this packet and authenticates the packet when it later receives the disclosed key. To continuously authenticate broadcast packets, μTESLA divides the time period for broadcast into multiple intervals, assigning different keys to different time intervals. All packets broadcast in a particular time interval are authenticated with the same key assigned to that time interval.

To authenticate the broadcast messages, a receiver first authenticates the disclosed keys. μTESLA uses a one-way key chain for this purpose. The sender selects a random value K_n as the last key in the key chain and repeatedly performs a (cryptographic) hash function F to compute all the other keys: $K_i = F(K_{i+1}), 0 \leq i \leq n-1$, where the secret key K_i

(except for K_0) is assigned to the i-th time interval. Because of the one-way property of the hash function, given K_j in the key chain, anybody can compute all the previous keys $K_i, 0 \leq i \leq j$, but nobody can compute any of the later ones $K_i, j+1 \leq i \leq n$. Thus, with the knowledge of the initial key K_0, which is called the *commitment* of the key chain, a receiver can authenticate any key in the key chain by merely performing hash function operations. When a broadcast message is available in the i-th time interval, the sender generates a MAC for this message with a key derived from K_i, broadcasts this message along with its MAC, and discloses the key K_{i-d} for time interval I_{i-d} in the broadcast message (where d is the disclosure lag of the authentication keys).

Each key in the key chain will be disclosed after some delay. As a result, the attacker can forge a broadcast packet by using the disclosed key. μTESLA uses a security condition to prevent such situations. When a receiver receives an incoming broadcast packet in time interval I_i, it checks the security condition $\lfloor (T_c + \Delta - T_1)/T_{int} \rfloor < i + d - 1$, where T_c is the local time when the packet is received, T_1 is the start time of the time interval 1, T_{int} is the duration of each time interval, and Δ is the maximum clock difference between the sender and itself. If the security condition is satisfied, i.e., the sender has not disclosed the key K_i yet, the receiver accepts this packet. Otherwise, the receiver simply drops it.

μTESLA is an extension to TESLA [Perrig et al. 2000]. The only difference between TESLA and μTESLA is in their key chain commitment distribution schemes. TESLA uses asymmetric cryptography to bootstrap new receivers, which is impractical for current sensor networks due to its high computation and storage overheads. μTESLA depends on symmetric cryptography (with the master key shared between the sender and each receiver) to bootstrap the new receivers individually. TESLA was later extended to include an immediate authentication mechanism [Perrig et al. 2001]. The basic idea is to include an image under a hash function of a late message content in an earlier message so that once the earlier message is authenticated, the later message content can be authenticated immediately after being received. This extension can also be applied to μTESLA.

B. PACKET DELIVERY RATES FOR MICA2 AND MICAZ IN INDOOR ENVIRONMENTS

We performed some in-door experiments to confirm the packet loss rate for MICAz with large packet sizes. We used both MICA2 and MICAz in our experiments for comparison purposes. The Radio Frequency (RF) module of MICA2 runs at frequency 916.7MHz, while that of MICAz runs at frequency 2.425GHz. We set the transmission power as $-10dbm$ on both MICA2 and MICAz. Figure 10 shows the comparison of packet delivery rates for MICA2 and MICAz when the packet payload size is 102 bytes (i.e., the maximum payload size in IEEE 802.15.4). It is easy to see that the packet delivery rate for MICAz remains above 95% in all test cases, when this rate quickly drops to 0 as the distance between the sender and the receiver increases from 50 feet to 90 feet. This results confirms our assumption that it is practical to have a large enough packet that can accommodate a digital signature on IEEE 802.15.4 compliant sensor nodes.

C. PROBABILITY OF FINDING A PUZZLE SOLUTION WITHIN 2^{L+C} TRIALS

The probability of finding a solution to a puzzle with strength l after x trials is

$$P_{x,l} = 1 - (1 - 2^{-l})^x.$$

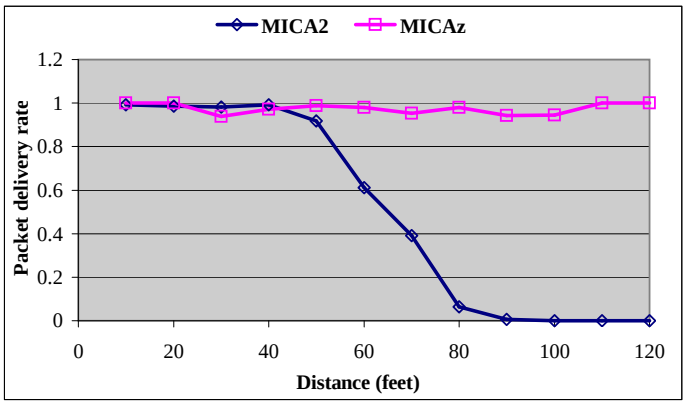

Fig. 10. Packet delivery rates for MICA2 and MICAz in our indoor experiments (Payload size: 102 bytes). Note that the proposed techniques are developed for ZigBee-compliant sensor nodes such as MICAz, not MICA2.

Given a $(l+c)$-bit buffer entry, where c is a constant positive integer, we can try at most $x = 2^{l+c}$ times for a puzzle solution. Thus, after $x = 2^{l+c}$ trials, the probability of finding a solution that can fit in the buffer entry becomes

$$P_l = 1 - (1 - 2^{-l})^{2^{l+c}}.$$

In the following, we prove that the probability P_l decreases as l $(l \geq 1)$ increases.

PROOF. We prove by showing $P_l' < 0$ when $l \geq 1$.
Let $f_l = 1 - 2^{-l}$, and $g_l = 2^{l+c}$. Then we have $P_l' = -(f_l^{g_l})'$. Since

$$\begin{aligned}(f_l^{g_l})' &= f_l^{g_l}(f_l' \cdot \frac{g_l}{f_l} + g_l' \cdot \ln(f_l)) \\ &= (1-2^{-l})^{2^{l+c}}(2^{-l} \cdot \ln 2 \cdot \frac{2^{l+c}}{1-2^{-l}} + 2^{l+c} \cdot \ln 2 \cdot \ln(1-2^{-l})) \\ &= (1-2^{-l})^{2^{l+c}} \cdot \ln 2 \cdot 2^{l+c} \cdot (\ln(1-2^{-l}) + \frac{1}{2^l - 1}),\end{aligned}$$

we have $P_l' = -(f_l^{g_l})' = -(1-2^{-l})^{2^{l+c}} \cdot \ln 2 \cdot 2^{l+c} \cdot (\ln(1-2^{-l}) + \frac{1}{2^l-1})$.

We need to show $P_l' < 0$ when $l \geq 1$. Because $l \geq 1$, we can easily have $(1-2^{-l})^{2^{l+c}} \cdot \ln 2 \cdot 2^{l+c} > 0$. For convenience, let $h_l = \ln(1-2^{-l}) + \frac{1}{2^l-1}$. We can determine $h_l > 0$ when $l \geq 1$, because

$$h_l' = (\ln(1-2^{-l}) + \frac{1}{2^l-1})' = \frac{2^{-l} \cdot \ln 2}{1-2^{-l}} - \frac{2^l \cdot \ln 2}{(2^l-1)^2} = \frac{-\ln 2}{(2^l-1)^2} < 0,$$

and

$$\begin{aligned}\lim_{l\to\infty} h_l &= \lim_{l\to\infty}(\ln(1-2^{-l}) + \frac{1}{2^l-1}) = \lim_{l\to\infty}(1 + \frac{\ln(1-2^{-l})}{\frac{1}{2^l-1}}) \\ &= \lim_{l\to\infty}(1 + \frac{\frac{2^{-l}\cdot\ln 2}{1-2^{-l}}}{-\frac{2^l\cdot\ln 2}{(2^l-1)^2}}) = 1 - \lim_{l\to\infty}\frac{2^l-1}{2^l} = 0.\end{aligned}$$

Thus, $P_l' = -(f_l^{g_l})' = -(1-2^{-l})^{2^{l+c}} \cdot \ln 2 \cdot 2^{l+c} \cdot (\ln(1-2^{-l}) + \frac{1}{2^l-1}) < 0$. In other words, P_l decreases when l ($l \geq 1$) increases. □

Note that P_l may still be very close to 1 when c is a positive integer. For example, when $l = 128$ and $c = 6$, $P_{l=128} = 1 - 1.6 \times 10^{-28}$. This implies that when $c = 6$, the probability of finding a solution with up to $l + 6$ bits for a puzzle with strength $l < 128$ is at least $1 - 1.6 \times 10^{-28}$.