Pre-Authentication Filters: Providing DoS Resistance for Signature-Based Broadcast Authentication in Sensor Networks

Qi Dong Donggang Liu[*]
iSec Laboratory
Department of Computer Science and Engineering
The University of Texas at Arlington
Arlington, TX 76019-0015
{qi.dong,dliu}@uta.edu

Peng Ning[†]
Cyber Defense Laboratory
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
pning@ncsu.edu

ABSTRACT

Recent studies have demonstrated that it is possible to perform public key cryptographic operations on the resource-constrained sensor platforms. However, the significant resource consumption imposed by public key cryptographic operations makes such mechanisms easy targets of Denial-of-Service (DoS) attacks. For example, if digital signatures such as ECDSA are used directly for broadcast authentication without further protection, an attacker can simply broadcast forged packets and force the receiving nodes to perform a large number of unnecessary signature verifications, eventually exhausting their battery power. This paper studies how to deal with such DoS attacks when signatures are used for broadcast authentication in sensor networks. In particular, this paper presents two filtering techniques, a *group-based filter* and a *key chain-based filter*, to handle DoS attacks against signature verification. Both methods can significantly reduce the number of unnecessary signature verifications that a sensor node has to perform. The analytical results also show that these two techniques are efficient and effective for resource-constrained sensor networks.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General-Security and protection

General Terms

Security, Design, Algorithms

[*]Liu's work is supported by the Army Research Office (ARO) under grant W911NF-07-1-0227.

[†]Ning's work is supported by the National Science Foundation (NSF) under grants CAREER-0447761 and CNS-0430223, and by ARO under grant W911NF-05-1-0247.

Keywords

Sensor Networks, Security, Broadcast Authentication, DoS Attacks

1. INTRODUCTION

A wireless sensor network typically consists of a large number of resource-constrained sensor nodes and possibly a few powerful control nodes (called *base stations*) [1]. These nodes usually communicate and collaborate with their neighbor nodes through low-power wireless links, and provide fine-grained sensing of physical conditions (e.g., temperature) from their immediate surroundings. The unique features of sensor networks lead to many attractive applications, such as target tracking and battlefield surveillance.

Many applications require sensor networks to operate correctly even in hostile environments. Security thus becomes a critical requirement for network protocols. Due to the sheer number of sensor nodes and the broadcast nature of wireless links, it is often desirable for a base station to broadcast commands and data to the network. The authenticity of such commands and data is critical for the correct operation of sensor networks. If convinced to accept forged or modified commands and data, sensor nodes may perform unnecessary and incorrect operations, and cannot fulfill the intended purposes of the network.

There are in general two types of solutions for broadcast authentication in sensor networks, μ*TESLA* [19] and *digital signature* [5]. μTESLA and its variations achieve broadcast authentication through delayed disclosure of authentication keys. Its efficiency is based on the fact that only symmetric cryptographic operations are needed to authenticate a broadcast message. Despite the efficiency, μTESLA has some undesirable features such as the need for (loose) time synchronization and the authentication delay.

Recent studies have demonstrated that it is possible to perform public key cryptographic operations on resource-constrained sensor platforms [5]. In addition, there have been continuous efforts to optimize Elliptic Curve Cryptography (ECC) for sensor platforms [12, 22, 14]. We thus believe that ECC-based signature schemes will also be an attractive option for broadcast authentication in many applications. However, the significant resource consumption imposed by public key cryptographic operations makes such mechanisms easy targets of Denial-of-Service (DoS) attacks.

For example, if ECDSA is used directly for broadcast authentication without further protection, an attacker can simply broadcast forged packets and force the receiving nodes to perform a large number of unnecessary signature verifications, eventually exhausting their battery power.

Note that receiving packets also consumes the energy on sensor nodes. Hence, an adversary can also launch DoS attacks by sending many bogus packets to jam the channel and exhaust the energy on the victim nodes. However, this is significantly less efficient than the DoS attacks against signature verification. Suppose every packet has 102 bytes payload [8]. A MICAz mote will transmit 133 bytes in the physical layer [21], which will cost the receiving node about $133 \times 8/250,000 = 4.256$ms to receive. On the other hand, verifying a 40-byte ECDSA signature takes about 1.96 seconds [22]. As indicated in [3], an active MICAz CPU will cost about two fifth of the energy of receiving packets for the same period of time. This indicates that the DoS attacks against signature verification is about 184 times more efficient than the simple jamming attack.

In this paper, we propose to apply *pre-authentication filters* to remove bogus messages before the actual signature verification is performed. Specifically, we develop two filtering techniques, a *group-based filter* and a *key chain-based filter*, to help sensor nodes avoid performing many unnecessary signature verifications. Both methods take advantage of the fact that broadcast in sensor networks is usually done through a network-wide flooding protocol, and a broadcast message from a sensor node usually has a small number of immediate receivers due to the low-power, short-range radio used in wireless sensor networks.

The proposed pre-authentication filters provide complementary capabilities in dealing with DoS attacks against signature-based broadcast authentication. The group-based filter organizes the neighbor nodes of a (local) sender into multiple groups, which are protected by different keys organized in a tree structure. Using these group keys, this mechanism not only facilitates the neighbor nodes to filter out forged messages, but also helps the sender adaptively isolate compromised nodes that launch DoS attacks. Unfortunately, the group-based filter allows compromised nodes to send forged messages before they are isolated.

The key chain-based filter employs a *two-layer* method, completely preventing compromised neighbor nodes from affecting benign ones. The first layer uses one-way key chains to mitigate the DoS attacks against signature verification, and the second layer uses pairwise keys to mitigate the DoS attacks on the verification of the chained keys in the first layer. However, despite the advantage in tolerating compromised nodes, the key chain-based filter defers to the group-based filter in the ability to tolerate packet losses.

Our analytical results show that both group-based and key chain-based filters can efficiently and effectively thwart the DoS attacks on signature verification.

The remainder of this paper is organized as follows. Section 2 discusses system and adversary models. Section 3 presents the proposed filtering techniques for signature verification in sensor networks. Section 4 reviews related work on sensor network security. Section 5 concludes this paper and points out several future research directions.

2. SYSTEM AND ADVERSARY MODELS

In this paper, we assume that a sensor node is able to do public key cryptographic operations such as signature generation and verification once for a while. We assume that a public key crypto-system such as ECC [5] has been employed in the network, and broadcast authentication is simply achieved by digitally signing every broadcast message. We consider a simple case of broadcast authentication where a base station needs to broadcast messages to the entire network in an authenticated manner. In addition, the keying materials (i.e., the public key) needed for signature verification have been distributed to all nodes in the network. The main design goal in this paper is *to effectively mitigate the DoS attacks against signature verification*.

An attacker can launch a wide range of attacks against the network. For example, he can simply perform DoS attacks to block the wireless channel. Such DoS attacks are simple but common to the protocols in sensor networks; there are no effective ways to stop an attacker from mounting such attacks. This paper thus focuses on the attack *whose goal is simply to fool sensor nodes to perform a significant number of unnecessary signature verifications*. We assume that the attacker can eavesdrop, modify, forge, replay or block any network traffic. We also assume that the attacker can compromise a few sensor nodes and learn all the secrets (e.g., cryptographic keys) on the compromised nodes [6].

3. PRE-AUTHENTICATION FILTERS

As we discussed earlier, when digital signatures such as ECDSA are used for broadcast authentication, an adversary can forge signatures and fool sensor nodes to perform a substantial number of expensive but unnecessary signature verifications. To deal with such DoS attacks, we propose to use *hop-by-hop pre-authentication filters* to remove bogus messages before verifying the actual digital signatures.

We develop two filtering techniques, a *group-based filter* and a *key chain-based filter*. Both methods take advantage of the fact that broadcast in sensor networks is usually done by a network-wide flooding protocol [11, 18], and a flooding message from a sensor node only has a small number of receivers due to the low-power, short-range radio. Since both filters are independent from the broadcast protocol, we simply assume a flooding scheme for broadcast and will not discuss how it is achieved. We will focus on the situation where *a sensor node needs to re-broadcast a digitally signed message to its neighbors*.

3.1 Group-Based Filter

A simple method to filter out forged messages is to authenticate the broadcast message, in a "hop-by-hop" manner, with a (local) group key shared among a (local) sender and its neighbor nodes. As a result, an adversary will not be able to forge messages without compromising the group key. However, sensors could be captured [6], which allows an adversary to forge as many messages as he wants using the group key on the compromised nodes. Alternatively, a sensor node can add a message authentication code (MAC) to a broadcast message for each of its neighbor nodes. However, this incurs large communication overhead even for a moderate neighbor size.

The above two simple ideas represent two extreme cases. One achieves high efficiency (only one MAC for every message), but is vulnerable to a single compromised neighbor; the other achieves high security, but introduces high communication overhead. In this subsection, we present a *group-*

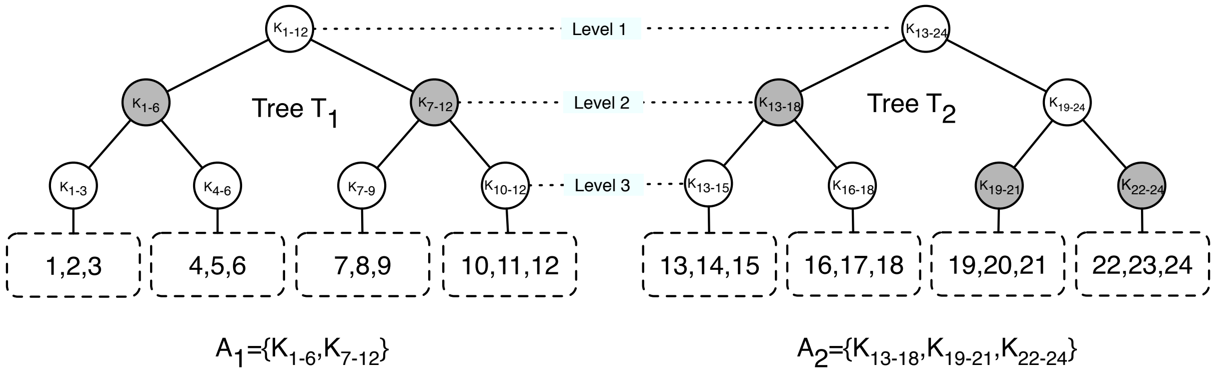

Figure 1: Example of key trees when $m = 2$ and $L = 2$. Assume neighbor size $b = 24$

based method to trade-off communication efficiency with security. Specifically, every sender divides its neighbor nodes into multiple groups. Every group uses a group key for pre-authentication filtering. When a group key becomes suspicious, we divide the corresponding group into smaller groups to isolate suspicious nodes.

3.1.1 Protocol Description

The group-based filter consists of three steps: *initialization*, *broadcasting*, and *re-keying*. In the first step, a (local) sender pre-loads keying materials to its neighbor nodes. In the second step, the sender re-broadcasts authenticated messages. In the last step, the sender re-selects keys dynamically to deal with compromised neighbor nodes.

In this paper, we assume the existence of a pairwise key establishment protocol in the network. This protocol can help every pair of sensor nodes to setup a secret key to protect their communication. A number of key pre-distribution protocols can be used for our purpose [4, 2, 13]. We let $K_{u,v}$ denote the shared pairwise key between nodes u and v. We assume that each node can add m MACs into a broadcast message, where each MAC is q-bit long.

Initialization: For each sender node u, let $N(u)$ be the set of its neighbors. u first divides $N(u)$ into m equal-sized groups, $\{g_1, ..., g_m\}$. We assume each group g_i has s nodes. For each g_i, we further divide it into 2^L equal-sized sub-groups, called *unit groups*, where L is a system parameter that determines the number of bits allocated for a unit group for filtering purposes. (Indeed, it is $\frac{q}{2^L}$ bits). For example, in Figure 1, we have two groups $g_1 = \{1, 2, ..., 12\}$ and $g_2 = \{13, 14, ..., 24\}$. Each of them is further split into 4 unit groups with 3 nodes in each unit group.

For each g_i, we construct a full binary tree T_i from the 2^L unit groups with each leaf representing a unit group. Each node in T_i is assigned a unique and random key. The resulting tree is called the *key tree*. (In total, we have m key trees for every sender.) The keying materials for each neighbor in group g_i includes the keys on the path from the corresponding unit group to the root of key tree T_i. For example, in Figure 1, node 16 belongs to g_2 and has three keys K_{16-18}, K_{13-18} and K_{13-24}.

Each key k in the j-th level of T_i defines a *level-j group*, which includes all nodes that know this key. Hence, every g_i is a level-1 group, and every unit group is a level $L + 1$ group. Let $l(k)$ be the level number of the group defined by key k in its key tree. The key tree T_i is used for filtering the messages sent to the nodes in g_i. We only use a subset A_i of the keys in T_i for filtering. This set, called *active key set*, is initialized as only including the root of T_i, and adjusted (in the third step) based on the suspiciousness of the nodes in g_i. In general, every path from a leaf to the root covers exactly one key in this set. The purpose is to make sure that every node in g_i will be able to find a piece of information in A_i to filter out forged messages. For example, in Figure 1, $A_2 = \{K_{13-18}, K_{19-21}, K_{22-24}\}$. This key set is picked because K_{13-24} and K_{19-24} were reported to be suspicious. We can see that every path from a leaf to the root of T_2 includes exactly one key in the A_2.

Broadcasting: Assume node u has received an authenticated, *digitally signed message* M, and needs to re-broadcast it according to the flooding protocol. u will add m *commitment values* to this message, one for each group g_i. The commitment values are generated as follows.

Consider a given group g_i. For every $k \in A_i$, node u computes a *commitment fragment*, which is the first $\frac{q}{2^{l(k)}}$ bits of $H(k||M)$, where H is a one-way hash function and "$||$" is the concatenation operation. The fragments generated from all the keys in A_i are then concatenated together, producing a complete commitment value for the nodes in g_i. The overall length of this commitment value will be the same as the length of a MAC. For example, in Figure 1, $A_2 = \{K_{13-18}, K_{19-21}, K_{22-24}\}$. If $q = 64$, the corresponding commitment value includes the first 32 bits of $H(K_{13-18}||M)$, the first 16 bits of $H(K_{19-21}||M)$, and the first 16 bits of $H(K_{22-24}||M)$. Let W be the set of all the commitment values (there are m of them) generated from all key trees. Node u will simply broadcast $\{M, W\}$.

Suppose a neighbor v in group g_i receives $\{M, W\}$. It will first identify the fragment in W that is generated based on a key k it knows. The information needed for identification can be easily obtained from u, as we will show in the third step. If the fragment is the same as the first $\frac{q}{2^{l(k)}}$ bits of $H(k||M)$, node v will do the actual signature verification; otherwise, it will ignore the message.

When the signature verification succeeds, node v extracts M and returns M to the sensor application for various uses. For example, it may also decide to relay M based on the flooding protocol. When the signature verification fails, node v checks if the number of failed signature verifications exceeds a threshold τ during the last $w = 2^{\frac{q}{2^L}}$ forged messages, i.e., the messages that failed either pre-authentication filtering or signature verification. (We will discuss why we set w in this way in our later analysis.) If so, node v believes that k is suspicious. In this case, if $l(k) < L + 1$, node v will stop accepting any message from u and also notify u to adjust the corresponding active key set. If $l(k) = L + 1$, v will stop processing the next w messages from u before returning to normal.

Re-Keying: When the sender u receives a report that the key k of T_i is suspicious, if the level number $l(k) < L+1$, it will use the two keys corresponding to key k's two child nodes to replace key k in the active key set A_i. In other words, u splits the level $l(k)$ group defined by k into two smaller groups to isolate suspicious nodes. For example, in Figure 1, if K_{13-18} is found to be suspicious, we will use K_{13-15} and K_{16-18} to replace K_{13-18} in A_2. Sender u will also notify the affected neighbors about the splitting so that they are able to identify the correct fragments in a broadcast message for pre-authentication filtering.

3.1.2 Security Analysis

In our approach, a sensor node will not verify the signature unless the corresponding fragment is valid. When none of the sender u and the nodes in $N(u)$ is compromised, an adversary has to guess the keys on the roots of key trees. Given the hardness of inverting a one-way hash function, we know that it is computationally infeasible for the attacker to bypass pre-authentication filtering. This prevents the adversary from mounting DoS attacks against any of the sensor nodes in $N(u)$.

We now focus on the security when there are compromised sensor nodes. We will study the security of our approach in the following two cases: (1) the sender is benign, and (2) the sender is compromised. After the analysis, we will summarize some important conclusions about the proposed approach.

The group-based approach has many system parameters that need to be configured properly. The configuration of these parameters will be discussed during the analysis.

Security under Benign Sender: Consider a benign node u that receives an authenticated message and needs to re-broadcast it according to the flooding protocol. Assume that the adversary has compromised N_c nodes in $N(u)$. We note that if a level $L + 1$ key is compromised, the adversary can forge messages with correct fragments and disable the broadcast authentication at the sensor nodes who share such key. We are thus interested in *how many benign neighbors will be affected by compromised neighbors.* That is, the number of benign neighbors that from time to time stop processing messages relayed by sender u.

Note that the most effective way of attacking our protocol is to make sure that the compromised nodes belong to different unit (level $L+1$) groups since a single malicious node in a unit group can disable the broadcast authentication of the nodes in this group. Hence, the number of benign nodes affected is no more than

$$N_c \times (\frac{s}{2^L} - 1) \tag{1}$$

For example, when $L = 2$ and $s = 8$, the attacker can only affect no more than N_c benign neighbors. In addition, the

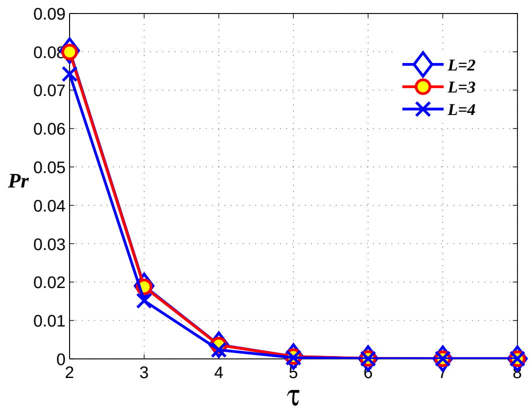

Figure 2: Probability p_r v.s. τ. Assume $q = 64$.

attacker has to continuously launch such attack since otherwise these affected nodes will return to normal situation after w messages from u. We also note that the number of affected benign nodes can be reduced by having a smaller s or larger L. However, having a small s leads to a larger m, and therefore more communication overhead. Having a larger L will reduce the security of our approach since the adversary is able to force nodes to do more unnecessary verifications, as we will see next.

When there are compromised neighbor nodes, it is possible that some non-compromised level $L + 1$ keys are used for pre-authentication filtering. When a benign node v is using a non-compromised level $L + 1$ key for pre-authentication filtering, we are interested in *the probability of an adversary bypassing pre-authentication filtering.* Since the length of a commitment fragment for a level $L + 1$ key is $\frac{q}{2^L}$-bit long, this probability can be estimated by

$$p_f = \frac{1}{2^{\frac{q}{2^L}}} \tag{2}$$

Equation 2 shows that a larger L may result in more unnecessary signature verifications at those benign nodes. For example, suppose $q = 64$, when $L = 3$, we have $p_f = \frac{1}{256} \approx 0.0039$; when $L = 4$, we have $p_f = 0.0625$. Together with Equation 1, we may want to find the largest value L that meets the required p_f.

When v uses a non-compromised level $L + 1$ key for pre-authentication filtering, we are also interested in *how likely v will stop processing messages from u when the adversary forges messages.* This is equivalent to the probability p_r of the adversary bypassing pre-authentication filtering τ times in a window of $w = 2^{\frac{q}{2^L}}$ forged messages. This probability can be estimated by

$$p_r = 1 - \sum_{i=0}^{\tau} \binom{w}{i} (\frac{1}{w})^i (\frac{w-1}{w})^{w-i} \tag{3}$$

Figure 2 shows that p_r decreases quickly with τ. Indeed, a small τ will make it very difficult for an attacker to disable the broadcast authentication at any benign node that has a non-compromised level $L + 1$ key. We also note that the value of L does not affect such probability much, making it much easier for us to configure threshold τ. This also explains our configuration of w in the protocol description.

There is a small probability that an adversary can disable the pre-authentication filter at a node with a non-compromised level $L+1$ key. However, this node will continue the protocol after w messages from the sender, which makes the attacker's job much harder. We can thus conclude that the adversary cannot gain much benefit by attacking those having non-compromised level $L+1$ keys.

Security under Compromised Sender: When the sender u is compromised, it can certainly forge broadcast messages with correct commitment values. We consider the worst case scenario where the sender u keeps forging broadcast messages but the number of failed verifications in any window of $2^{\frac{q}{2^L}}$ forged messages is always no more than τ. In this case, obviously, all the neighbor nodes will keep processing the messages from node u and verifying the corresponding signatures. Fortunately, due to threshold τ, we can clearly see that the fraction p_f of forged messages that will lead to unnecessary signature verifications is always bounded by

$$p_f = \frac{\tau}{2^{\frac{q}{2^L}}} \tag{4}$$

For example, when $q = 64$, $L = 3$ and $\tau = 3$, we have $p_f = 0.012$. We can see that our group-based approach can effectively mitigate the DoS attacks launched by a compromised sender given a reasonable setting of parameters.

Summary: First of all, whenever a benign node receives a true broadcast message from a benign sender, it will never miss this true message as long as the pre-authentication filter is not disabled due to a compromised neighbor node in the same level $L+1$ group. The impact of compromised nodes is given by Equation 1. For the forged messages comes from an adversary who may compromised the sender and some neighbor nodes, we have the following theorem.

THEOREM 1. *The fraction of forged messages that can lead to unnecessary signature verifications at a benign node is no more than*

$$\frac{(1-f_c)^{\frac{s}{2^L}-1} + \tau \times [1-(1-f_c)^{\frac{s}{2^L}-1}]}{2^{\frac{q}{2^L}}},$$

where f_c is the fraction of compromised sensor nodes.

PROOF. Note that the size of a level $L+1$ group is $\frac{s}{2^L}$. Hence, for a given neighbor node, the probability that the sender and all other nodes in the same level $L+1$ group are benign can be estimated by $(1-f_c)^{\frac{s}{2^L}-1}$. In this case, the fraction of forged messages that will lead to unnecessary signature verifications is given by Equation 2. When either the sender or one of the nodes in the same level $L+1$ group is compromised, the fraction of forged messages that will lead to unnecessary signature verifications is given by Equation 4. Overall, the fraction of forged messages that will lead to unnecessary signature verifications at a benign node is no more than

$$\frac{(1-f_c)^{\frac{s}{2^L}-1} + \tau \times [1-(1-f_c)^{\frac{s}{2^L}-1}]}{2^{\frac{q}{2^L}}}.$$

□

Theorem 1 indicates the security of our group-based approach in dealing with the DoS attacks against signature verification. Since L, s and τ are usually quite small, we can see that our group-based filter can effectively defeat the DoS attacks against signature verification. For example, when $L = 3, s = 16, \tau = 3$, $q = 64$ and $f_c = 0.1$, the fraction of forged messages leading to unnecessary verifications is only about 0.0047.

3.1.3 Overheads

According to the protocol description, every sensor node needs to store a set of key trees (as a sender), and $L+1$ keys for every neighbor (as a receiver). Since the keys on each key tree can be derived from a random seed, they only need one master key. Hence, the storage overhead can be estimated by $b \times (L+1) \times q$ bits, where b is the average neighbor size in the network.

The communication overhead introduced by our protocol comes from three parts: (1) the distribution of keying materials in the initialization, (2) the reports from every neighbor node indicating unsafe keys and the notifications to neighbor nodes indicating the change of active key sets, and (3) the space needed for the commitment values in broadcast messages. The distribution of keying materials only happens during the initialization; it only involves the delivery of $L+1$ keys for every neighbor nodes and only needs to be done once for each neighbor. For the reports and notifications, we note that they are done between neighbors and the number of messages is usually limited. Hence, the main communication overhead introduced is the additional m MACs on every broadcast message.

We use MICAz motes to show the energy consumption for the additional MACs. With the 250kbps data rate and 3.0V power level, MICAz will keep the current draw at no more than 17.4mA in TX mode and 19.7mA in RX mode [3]. Assume $m = 8$ and $q = 64$. The energy cost of sending the additional MACs can be estimated as $3.0 \times 17.4 \times 64 \times 8/250,000 = 0.107$mJ, and the energy cost of receiving can be estimated as $3.0 \times 19.7 \times 64 \times 8/250,000 = 0.121$mJ. On the other hand, the current draw of an active MICAz CPU is 8mA [3]. As discussed before, verifying an ECDSA signature takes about 1.96s on MICAz motes. Thus, the energy cost of a signature verification can be estimated as $3.0 \times 8 \times 1.96 = 47.04$mJ, which is about 400 times more than the energy cost of sending or receiving the additional MACs. This clearly explains the benefit of our filtering method.

According to the protocol description, each sender needs to do up to $m \times 2^L$ additional hash operations on average. For each receiver, it needs to perform one additional hash operation to verify the corresponding fragment.

3.1.4 Extension: Adaptive Re-Grouping

Our previous analysis indicates that we usually prefer a small L to make sure that an adversary cannot fool sensor nodes to perform many unnecessary signature verifications. However, we also mentioned that a smaller L will make it possible for a single compromised sensor node to disable the broadcast authentication at more benign nodes.

In the following, we present an *adaptive re-grouping* extension to deal with compromised neighbor nodes and provide more flexibility in configuring parameter L. The basic observation is that benign nodes will always report suspicious keys in the key trees to the sender. This provides evidence for the sender to reason about the suspiciousness of neighbor nodes. The sender can then rank and group neighbor nodes according to the suspiciousness, making sure that *a benign* *neighbor is likely to be in the same group with other benign neighbors and a compromised neighbor is likely to be in the same group with other compromised neighbors.* Achieving this will makes it much more difficult for an adversary to impact benign sensor nodes.

Adaptive Re-Grouping: We focus on level $L+1$ keys. When a level $L+1$ key is found to be unsafe, all nodes having this key are suspicious. We consider them as equally suspicious. A sensor node will send a report to the sender when its level $L+1$ key is found to be unsafe. The adaptive re-grouping protocol works as follows.

The sender u maintains a boolean variable $c(i)$ for every neighbor node $i \in N(u)$, indicating if this neighbor node is suspicious. Initially, we have $c(i) = FALSE$ for every $i \in N(u)$. Whenever the sender u receives a report from its neighbor node i saying that its level $L+1$ key is suspicious, node u will set $c(j) = TRUE$ for every neighbor node j in the same level $L+1$ group as node i.

The sender u periodically re-groups neighbor nodes. During each round of re-grouping, node u first classifies all current level $L+1$ groups into two categories: the first category includes those with non-suspicious level $L+1$ keys, and the second category includes those with suspicious level $L+1$ keys. This classification can be done using the boolean variables $\{c(i)\}_{i \in N(u)}$. The sensor nodes in the second category are then re-grouped randomly, and we expect that some of these randomly organized level $L+1$ groups only include benign nodes so that they will be identified as benign with a very high probability in the next round. In this way, we will build more and more benign level $L+1$ groups.

After re-grouping, sender u then re-generates random keys for every key tree. After the assignment of new keys, we will have new key trees for the new group construction. The sender will then distribute new keying materials to neighbor nodes and reset all the boolean values $\{c(i)\}_{i \in N(u)}$ to FALSE.

Advantages: The total number of level $L+1$ groups can be estimated by $G = m \times 2^L$. Consider the moment right before the $i+1$-th re-grouping at sender u. Let m_i be the number of level $L+1$ groups that include only benign neighbors. For each of the groups, the probability of being marked as suspicious after the $i+1$-th re-grouping can be estimated by Equation 3. Thus, on average, these m_i groups in the first category will contribute $m_i(1-p_r)$ to m_{i+1}.

During the re-grouping, the sender will believe that there are up to $(G - m_i(1-p_r))$ suspicious level $L+1$ groups. Assume there are N_c compromised neighbors. After the $i+1$-th re-grouping, the average number of level $L+1$ groups in the second category that include only benign nodes can be estimated by

$$(G - m_i(1-p_r))(\frac{G - m_i(1-p_r) - 1}{G - m_i(1-p_r)})^{N_c}.$$

Overall, after the $i+1$-th re-grouping, the average number of level $L+1$ groups that include only benign sensor nodes (m_{i+1}) can be estimated by

$$m_i(1-p_r) + (G - m_i(1-p_r))(\frac{G - m_i(1-p_r) - 1}{G - m_i(1-p_r)})^{N_c}.$$

Figure 3 shows the performance of our re-grouping approach. We consider the worst scenario where the compromised nodes belong to different level $L+1$ groups. Hence, $m_0 = G - N_c = m \times 2^L - N_c$. Clearly, without re-grouping, the number of affected groups is N_c. With re-grouping, we can clearly see from the figure that we significantly increase the average number of unaffected level $L+1$ groups.

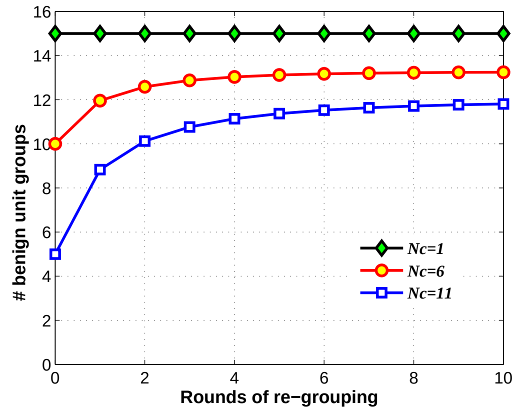

Figure 3: Performance of adaptive re-grouping ($L = 3$, $\tau = 3$ and $m = 2$).

Overheads: The proposed re-grouping idea does not add much overhead to sensor nodes. Indeed, for every neighbor node, it only needs to report at most one extra message and get re-initialized after a certain period of time. There are no other additional overheads for them. For the sender, it needs to collect at most one report from each neighbor node and re-initialize the keying materials after a certain period of time. In addition, it needs to maintain a boolean value for each level $L+1$ group, re-organize category-2 groups, and re-generate all keying materials. These additional steps will not add too much cost for the sender.

3.2 Key Chain-Based Filter

Due to the usage of group keys in the previous method, compromising a neighbor allows an attacker to bypass the pre-authentication filter at many benign nodes and fool them to do unnecessary verifications. This is one major issue for the previous method. In the following, we give another pre-authentication filter that doesn't offer an adversary much benefit by compromising neighbors.

Our main idea is to introduce an *asymmetry* property so that a receiver (neighbor) can filter but not forge messages. One-way key chains meet such requirement. A one-way key chain $\{K_1, ..., K_n\}$ has the following property: from any K_j, all former keys can be derived by iteratively hashing this key ($K_i = H^{j-i}(K_j), 0 \leq i < j$), but none of the later keys can be computed. The hash image of the first key K_1 is called the *commitment* of this key chain. With the commitment, anybody can verify the authenticity of any later key by only performing hash operations.

With one-way key chains, a simple pre-authentication filter (used by LHAP [25]) works as follows: the sender always adds the next unreleased key in the key chain into a new broadcast message. When a neighbor receives any message, it will perform the actual signature verification only when the chained key included in the message is found to be new to this neighbor. A long key chain is often used to support network operations continuously. However, an adversary may claim a key close to the end of key chain and fool a node to perform a large number of unnecessary hash operations, causing a DoS attack.

We apply a *two-layer filter* to deal with the DoS attacks on the verification of signatures and chained keys. The first layer employs a one-way key chain to filter out fake signatures, and the second layer uses existing pairwise keys to prevent a node from doing a significant number of unnecessary hash operations. Our approach is different from LHAP in that we have an additional layer of protection to stop attacks on the verification of chained keys.

The main intuition of the second layer is to *control the number of hash operations used for verifying a chained key.* A receiver (neighbor) will verify a chained key only when either the verification costs only a few hash operations or there is additional evidence (i.e., a *commitment value* generated from the shared key with the sender) about the authenticity of the message. Our design guarantees that a receiver can always find additional evidence in a small number of consecutive messages. Hence, when a receiver realizes that it has to do many hash operations to verify the key, it chooses to wait for additional evidence before doing the verification. This significantly enhances the security against the DoS attacks on the verification of chained keys since the attacker has to guess the pairwise key and then forge the evidence. After a successful verification, this receiver can immediately catch up with the sender since it knows the most recently released key in the key chain.

3.2.1 Protocol Description

The key chain-based filter consists of three steps, *initialization*, *broadcasting*, and *re-keying*. We assume that every node can add m MAC values and a sequence number into every broadcast message, where each MAC value is q-bit long.

Initialization: During the initialization, each node u first discovers a set $N(u)$ of neighbors and generates a one-way key chain of length n: $\{K_1, \cdots, K_n\}$. It then distributes the key chain commitment $K_0 = H(K_1)$ to every v in $N(u)$ via a secure channel established by the pairwise key $K_{u,v}$.

Sender u also maintains a variable idx to record the index of the next key in the key chain. Initially, we have $idx = 1$. This variable decides which authentication key in the key chain to use. In addition, u also organizes its neighbors into disjoint groups and picks one group for use in every round of broadcast authentication in *a round-robin manner*. These groups are used in the second layer of pre-authentication filter. For convenience, we call them *layer-2 authentication groups*. The value of idx directly determines which group to use for each broadcast message.

When a layer-2 authentication group is picked for a broadcast message, every node in this group can find additional evidence (a commitment value) in the message to do the filtering. Let l be the length of a commitment value. Since the first layer filtering needs space for one key, we have $q(m-1)$ bits left for additional values. Hence, we can add $\frac{q(m-1)}{l}$ commitment values in a broadcast message. The size of an authentication group is $\frac{q(m-1)}{l}$ (the last group may have fewer nodes).

Broadcasting: Assume node u has received an authenticated, digitally signed message M and needs to re-broadcast it according to the flooding protocol. Node u will first control the broadcast rate. Specifically, it will ensure that there will be no more than one true broadcast message from itself per T seconds on average, where T is a system parameter that determines the maximum frequency of relaying digitally-signed broadcast messages.

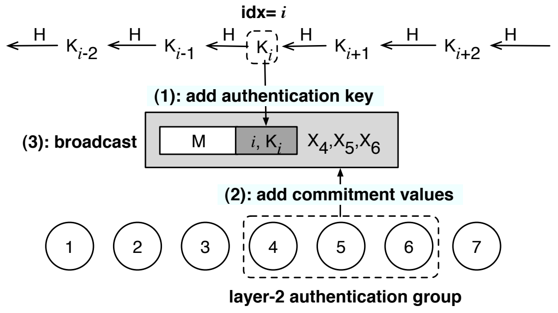

Figure 4: Example of the key chain-based approach. X_j is the first l bits of $H(M||i||K_i||K_{u,j})$.

After rate-controlling, node u will get the next key K_{idx} from the key chain and compute a set W of commitment values. To generate W, node u will first select the next layer-2 authentication group based on index idx. For every node v in this group, node u uses the first l bits of $H(M||idx||K_{idx}||K_{u,v})$ as the commitment value. The set W includes all commitment values generated in this way. Node u will then broadcast $\{M, idx, K_{idx}, W\}$ and increment the variable idx by one.

Figure 4 shows an example of re-broadcasting an authenticated, digitally-signed message M. In the example, node u has seven neighbors $\{1, ..., 7\}$, which are organized into three layer-2 authentication groups, $\{1, 2, 3\}$, $\{4, 5, 6\}$ and $\{7\}$. Node u first adds i and K_i to the message, and then computes and adds the commitment values for the second layer-2 authentication group $\{4, 5, 6\}$ to the message. The final broadcast packet includes M, i, K_i, and three commitment values $\{X_4, X_5, X_6\}$.

Suppose a neighbor node v receives $\{M, idx, K_{idx}, W\}$ from sender u. It first checks if W includes the commitment value generated from the key $K_{v,u}$ shared with sender u. This checking can be easily done with the value idx included in the message, since the sender u always picks a layer-2 authentication group for use in a round-robin manner. Let K_i be the chained authentication key that node v stores in its memory. The result of this checking will lead to the following two cases.

- If W does not include the commitment value generated from the key $K_{v,u}$, node v will check whether $0 < idx - i \leq B$, where B is a threshold value that determines how many hash operations a node is willing to pay for verifying a chained key. If not, node v will ignore the message.
- If W does include the commitment value generated from the key $K_{v,u}$, node v will first check whether the included commitment value equals the first l bits of $H(M||idx||K_{idx}||K_{v,u})$. If not, node v will ignore the message; otherwise, it will further check whether $idx - i > 0$ (for *freshness*). If not, node v will ignore the message.

Once the message has passed the above checking, node v

will verify the key K_{idx} using K_i. If such verification fails, node v will check whether the verification of authentication keys has failed more than τ times during the last 2^l forged messages that include the commitment value for node v. If yes, node v will stop processing the next 2^l messages from node u before returning to normal.

When the verification of the chained key K_{idx} succeeds, node v also checks whether the sender discloses chained keys at a rate of no more than one key per T seconds. If not, node v considers sender u as compromised and then removes u from its neighbor set $N(v)$. Otherwise, node v will perform the actual signature verification and use the new key K_{idx} to replace K_i. When the signature verification succeeds, node v extracts M and returns M to the sensor application for various uses. For example, it may also decide to relay M based on the flooding protocol.

Re-Keying: After every round of re-broadcast, node u will check whether $idx >= n$. If so, node u will generate a new key chain and distribute the initial commitment of the new key chain to every neighbor node. In addition, it will also reset idx to 1. When a neighbor node receives the updated key chain commitment, it will set its copy of chained keys to this commitment for future authentication.

3.2.2 Security Analysis

In the key chain-based method, sensor nodes only verify signatures in those messages that have passed our two-layer filtering. Similar to the security analysis for the group-based scheme, we will evaluate the security of the key chain-based scheme in two cases: (1) the sender is benign, and (2) the sender has been compromised. After security analysis, we will also summarize some important conclusions.

The key chain-based approach has many system parameters that need to be configured properly. The configuration of these parameters will be discussed during the analysis.

Security under Benign Sender: When the sender is benign, we first consider the first layer of pre-authentication filtering, which is achieved by an one-way key chain. This layer of pre-authentication filtering guarantees the following property.

LEMMA 1. *Given a benign sender, the number of unnecessary signature verifications a benign receiver performs will not exceed the number of true broadcast messages.*

PROOF. The freshness requirement of the chained key guarantees that nobody can forge broadcast messages using undisclosed keys in the key chain. Thus, no matter how many neighbor nodes are compromised, the total number of failed signature verifications that a benign neighbor performs will never exceed the total number of authentication keys in the key chain. Since the sender is benign, the total number of unnecessary signature verifications a benign neighbor node needs to do will never exceed the total number of true broadcast messages from the sender. □

Lemma 1 clearly indicates that an adversary is not able to convince any benign sensor node to do a significant number of unnecessary signature verifications. However, as mentioned before, an adversary can fool a sensor node to do a large number of unnecessary hash operations, causing a DoS attack. In the following, we will study how our second layer of pre-authentication filtering addresses this problem.

The second layer of pre-authentication filtering at sender u appends commitment values for $\frac{q(m-1)}{l}$ neighbors in each broadcast message. When a neighbor v receives a forged broadcast message that does not include the commitment for node v, it will perform no more than B hash operations in verifying the chained key included in the message.

When the broadcast message does include the commitment value for node v, the probability that the adversary can successfully generate the correct commitment value is $\frac{1}{2^l}$. Node v will need to do one hash operation for an incorrect commitment value and up to $n+1$ hash operations for a correct commitment value (one for verifying the commitment value and n for verifying the chained key included in the message). Thus, for those forged broadcast messages that include the commitment values for node v, the average number of additional hash operations that node v has to do will not exceed $1 + \frac{n}{2^l}$. This tells us that we can simply set $n = (B-1) \times 2^l$ to make sure that a benign neighbor node will not perform more than B additional hash functions for each forged message on average. For example, when $l = 8$ and $B = 6$, we can set $n = 1,280$. As a result, in this paper, we always set

$$n = (B-1) \times 2^l \tag{5}$$

Equation 5 also shows how to configure parameters B, l and n. Parameter B usually needs to be large enough to make sure that the non-malicious message loss will not impact the protocol. In other words, a sensor node should be able to receive at least one of any B consecutive messages from a sender with a very high probability. We then set parameter n based on the storage and computation costs that a sender is willing to pay. For example, when $n = 900$, a sender can allocate space for 30 keys to save one key per every 30 keys in the key chain, $\{K_{30}, K_{60}..., K_{900}\}$, and allocate space for 30 keys to save the keys for immediate use, $\{K_{30j+1}, K_{30j+2}, ..., K_{30j+10}\}$. As a result, the sender will need to first generate the whole key chain (900 keys) and then generate the other 870 keys again during the pre-authentication filtering. Overall, the sender will allocate space for 60 keys for the key chain, and perform no more than two hash operations on average to find a key to use. In general, the storage overhead to save a key chain of length n is $2\sqrt{n}$. After configuring B and n, we can easily set l.

We also note that when the sender is benign, the adversary still has a small chance to bypass the second layer of pre-authentication filtering by randomly guessing the commitment value. If the number of messages that passed the layer-2 filtering but failed the layer-1 filtering is larger than τ during the last 2^l forged messages that contain the commitment value for a given sensor node, this node will stop accepting messages from the sender. We are thus interested in the probability p_r that *a benign sensor node stops accepting any broadcast message from a benign sender.* Since the probability of the adversary making a correct guess of the commitment value is about 2^{-l}, the probability p_r can be estimated by

$$p_r = 1 - \sum_{i=0}^{\tau} \binom{2^l}{i} (\frac{1}{2^l})^i (1 - \frac{1}{2^l})^{2^l - i} \tag{6}$$

Figure 5 shows that a small τ is sufficient to make p_r very low. Also note that l does not affect p_r much. We can thus easily configure τ to tolerate forged messages.

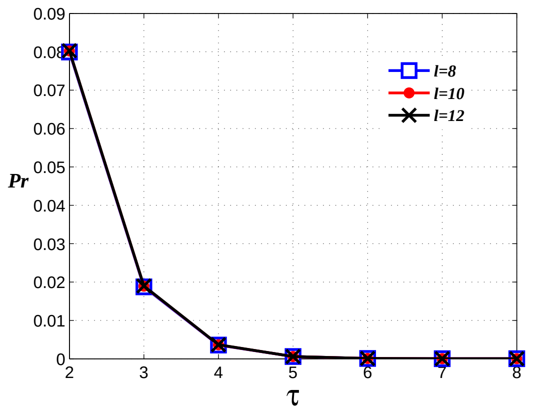

Figure 5: Probability p_r v.s. threshold τ.

Security under Compromised Sender: If the sender u is compromised, it can always forge broadcast messages with correct chained keys and commitment values. However, our approach can still effectively prevent a compromised sender from launching DoS attacks against signature verification by controlling the broadcast rate.

Lemma 2. *In case of a compromised sender, the average number of unnecessary signature verifications a benign node needs to do will never exceed one per T seconds.*

Proof. The proof of Lemma 1 indicates that the number of unnecessary signature verifications a benign neighbor needs to do will never exceed the number of authentication keys in the key chain. We also note that our approach controls the rate of disclosing chained keys to one per T seconds. Hence, the average number of unnecessary signature verifications a benign node needs to do will never exceed one per T seconds even if the sender is compromised. $\square$

Lemma 2 shows that a compromised sender is not able to convince any benign sensor node to do a significant number of unnecessary signature verifications. However, a compromised sender can fool a sensor node to perform a significant number of unnecessary hash functions for a forged message. Similarly, we will also study how our second layer of filtering addresses this problem.

Similar to the case of a benign sender, when a sensor node receives a forged broadcast message (from a compromised sender) that does not include the commitment value for itself, it will perform no more than B hash operations in verifying the chained key included in the message.

However, when the broadcast message does include the commitment value for the receiver, the adversary can always include a correct commitment since the compromised sender knows all the keys. Fortunately, due to the threshold τ, the compromised sender cannot force a benign node to do more than $n \times \tau$ additional hash operations in a window of 2^l forged messages that include the commitment of this node. The reason is that once the number of messages that failed the first layer filtering exceeds the threshold τ, the neighbor node will stop accepting messages from the compromised sender. Since n is set to $(B-1) \times 2^l$ (Equation 5), the average number of additional hash operations a benign neighbor needs to do will not exceed

$$\frac{n \times \tau}{2^l} + 1 = (B-1) \times \tau + 1 \qquad (7)$$

Summary: First of all, whenever a benign sensor node receives a true broadcast message from a benign sender, it has a very high chance of being able to authenticate this true message for a reasonable B. For the forged messages, we have the following two theorems.

Theorem 2. *On average, a benign sensor node will not do more than one unnecessary signature verifications in every T seconds for a given sender.*

The above theorem can be easily derived by generalizing Lemma 1 and Lemma 2. The detail of the proof will be thus skipped for simplicity. This theorem indicates the security performance of our key chain-based approach in dealing with the DoS attacks on signature verification.

Theorem 3. *The average number of additional hash operations a benign sensor node needs to do is no more than $B(1-f_c) + [(B-1)\tau + 1]f_c$, where f_c is the fraction of compromised sensor nodes.*

Proof. Note that when the sender is benign, the average number of additional hash operations will not exceed B. When the sender is compromised, the average number of additional hash operations will not exceed $(B-1)\tau+1$. Since the probability of any given sensor node being compromised is f_c, the average number of additional hash operations a benign sensor node needs to do will not exceed $B(1-f_c) + [(B-1)\tau+1]f_c$. $\square$

Theorem 3 indicates the security performance of our key chain-based filter in dealing with the DoS attacks on the verification of chained keys. Since B and τ are usually quite small, we can see that our key chain-based filter can effectively defeat this kind of DoS attacks.

3.2.3 *Overheads*

From the protocol description, any sensor node u needs to store a one-way key chain (as a sender) and a key chain commitment for every neighbor sender (as a receiver). Hence, the overall storage overhead can be estimated by $(b+2\sqrt{n}) \times q$ bits, where n is length of the key chain and b is the average neighbor size.

The communication overhead includes two parts: (1) the distribution of the key chain commitment, and (2) the space for the chained key and the commitment values in every broadcast message. Note that each key chain can be used for n broadcast messages. We thus believe that the communication overhead needed for distributing the initial key chain commitment will not be a problem given a reasonably long key chain. In addition, each broadcast message will include $m \times q$ bits additional information (i.e., the chained key and the commitment values).

For every broadcast message, a sender u will need to perform $|W|$ hash operations for generating the commitment values and an average of 2 hash operations for generating a key in the key chain to use. On the receiver side, Theorem 3 shows the average number of additional hash operations a benign sensor node needs to do. Overall, we can see that the key chain-based method is also effective and efficient.

3.3 Discussion

The detailed analysis on the security against DoS attacks and the overheads of the proposed two methods is given in the previous two subsections. In the following, we will discuss the security of these two schemes under some other common attacks and compare the proposed two methods in terms of security and overheads.

3.3.1 Security under Other Attacks

In the following, we will discuss the security of our two methods in the presence of *sybil* attacks [15], *wormhole* attacks [7], and *node replication* attacks [17].

Sybil Attacks: In sybil attacks, an adversary tries to clone sensor nodes with different IDs. However, some key pre-distribution techniques (e.g., [13]) can effectively remove the impact of sybil attacks since the keying materials for pairwise key establishment is bound with the node ID. When any of these key pre-distribution techniques is employed, we will be free from sybil attacks.

Wormhole Attacks: Wormhole attacks do impact the security of our proposed approaches since they can increase the size of neighbor nodes for a sensor node. However, such impact is very limited since a significant increase in the number of neighbor nodes usually indicates a wormhole attack. In this case, we can simply pick a subset of them for use in our protocol since as long as a sensor node can deliver its message to a sufficient number of neighbors, the broadcast protocol will work correctly.

Node Replication Attacks: By launching node replication attacks, the adversary can increase the fraction of compromised nodes in a local area. According to Theorem 1, we can see that this attack does impact our group-based approach. However, as we discussed, we can always configure parameters such as τ properly or re-grouping sensor nodes to mitigate the impact of increasing the fraction of compromised nodes. On the other hand, according to Theorem 2 and Theorem 3, increasing the fraction of compromised nodes does not generate much impact on the security of the key chain-based approach. Moreover, though the probability of finding a malicious sender is high, a node can always switch to other nodes if it notices that a particular sender is suspicious. Therefore, as long as the benign nodes in the network are well-connected, our protocols will work correctly.

3.3.2 Comparison

Both pre-authentication filters have advantages and disadvantages when compared with each other. In terms of security, the main difference between them is that the group-based approach allows a compromised neighbor node to disable the broadcast authentication at a number of other benign neighbor nodes even if the sender is benign. In contrast, the key chain-based approach does not have this problem. Thus, the key chain-based method can often perform better than the group-based approach when there is a large fraction of compromised sensor nodes. In addition, according to Theorem 2, we know that the key chain-based approach can have better security when there is a low rate of true broadcast messages from the base station. However, when there is only a small fraction of compromised sensor nodes, the group-based approach can be more secure in the sense that the fraction of fake signatures leading to unnecessary signature verifications can be made very small. In contrast, according to Theorem 2, we know that even without any compromised node, the key chain-based approach allows the adversary to fool sensor nodes to perform many unnecessary signature verifications if there is a high rate of true broadcast messages from the base station.

In terms of overheads, the group-based method performs slightly better than the key chain-based approach. First, the storage overhead of the group-based method ($b \times (L+1)$ keys) is comparable to that of the key chain-based method ($b + 2\sqrt{n}$ keys). Second, the communication cost of the group-based method only involves the initial distribution of keying materials and up to $L+1$ reports from every neighbor node to the sender. In contrast, the key chain-based method has to update the keys at neighbor nodes once for a while. Finally, the group-based approach requires up to $m \times 2^L$ hash operations for a sender and one hash operation for a receiver. In contrast, the key chain-based approach requires $\frac{(m-1)q}{l}+2$ hash operations for a sender, which is comparable to the group-based method, and $B(1-f_c)+[(B-1)\tau+1]f_c$ hash operations for a receiver, which can be many more than the group-based approach since B has to be configured to accommodate the lossy channel.

4. RELATED WORK

This section reviews some related work on sensor network security. A number of techniques have been developed for key management in sensor networks [4, 2, 13]. To build trustworthy sensor networks, researchers have studied methods to protect network services such as data aggregation [20], routing [9] and location discovery [10]. Many techniques have been proposed to detect wormhole attacks [7] and node replication attacks [17].

DoS attacks against sensor networks have been thoroughly evaluated in [24]. The DoS attacks against signature-based broadcast authentication have also been studied in [16, 23]. A weak authentication mechanism using cryptographic puzzles is proposed in [16] to reduce the number of false signature verifications. However, it requires a powerful sender and introduces the sender-side delay. The dynamic window scheme proposed in [23] can control the propagation of fake messages by making smart choices between verifying a message before forwarding it and forwarding a message before verifying it. However, a fake message will be propagated in the network until the victim node verifies the signature. This paper uses pre-authentication filters to efficiently and effectively filter out fake signatures before verifying them. This is a useful service that is complementary to the above studies.

5. CONCLUSION AND FUTURE WORK

ECC-based signature schemes have attracted a lot of attention recently for broadcast authentication in sensor networks. However, such approaches are vulnerable to DoS attacks against signature verifications. This paper shows how the proposed group-based and key chain-based methods effectively mitigate such DoS attacks.

In the future, we are particularly interested in developing our systems on real sensor platforms. It is also highly desirable to evaluate the performance of our approaches through field experiments to obtain more useful results such as the energy savings under attacks. In addition, we will also seek efficient solutions in scenarios where a sender's signal range can reach all or most of the sensor nodes in the network.

6. REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, 2002.

[2] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 197–213, May 2003.

[3] Crossbow Technology Inc. MICAz 2.4GHz Wireless Module. `http://www.xbow.com/Products/productdetails.aspx?sid=164`. Accessed in January 2008.

[4] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 41–47, November 2002.

[5] N. Gura, A. Patel, and A. Wander. Comparing elliptic curve cryptography and rsa on 8-bit CPUs. In *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, August 2004.

[6] C. Hartung, J. Balasalle, and R. Han. Node compromise in sensor networks: The need for secure systems. Technical Report CU-CS-990-05, U. Colorado at Boulder, Jan. 2005.

[7] Y. Hu, A. Perrig, and D. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings of INFOCOM*, April 2003.

[8] IEEE Computer Society. 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). *IEEE Std 802.15.4-2003*, 2003.

[9] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *Proceedings of 1st IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.

[10] L. Lazos and R. Poovendran. Serloc: Secure range-independent localization for wireless sensor networks. In *ACM workshop on Wireless security (ACM WiSe 2004)*, Philadelphia, PA, October 1 2004.

[11] H. Lim and C. Kim. Multicast tree construction and flooding in wireless ad hoc networks. In *Proceedings of ACM Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, 2000.

[12] A. Liu and P. Ning. TinyECC: Elliptic curve cryptography for sensor networks. `http://discovery.csc.ncsu.edu/software/TinyECC/index.html`.

[13] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)*, pages 52–61, October 2003.

[14] D. J. Malan, M. Welsh, and M. D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *Proceedings of First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON 2004)*, pages 71–80, 2004.

[15] J. Newsome, R. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: Analysis and defenses. In *Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN 2004)*, Apr 2004.

[16] P. Ning, A. Liu, and W. Du. Mitigating dos attacks against broadcast authentication in wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(1), 2008. To appear.

[17] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *IEEE Symposium on Security and Privacy*, May 2005.

[18] W. Peng and X. Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Proceedings of ACM International Symposium on Mobile and Ad Hoc Networking and Computing*, 2000.

[19] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. SPINS: Security protocols for sensor networks. In *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks (MobiCom)*, July 2001.

[20] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Nov 2003.

[21] Texas Instruments Inc. 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. `http://focus.ti.com/lit/ds/symlink/cc2420.pdf`. Accessed in January 2008.

[22] H. Wang, B. Sheng, C. C. Tan, and Q. Li. WM-ECC: an Elliptic Curve Cryptography Suite on Sensor Motes. Technical Report WM-CS-2007-11, College of William and Mary, Computer Science, Williamsburg, VA, 2007.

[23] R. Wang, W. Du, and P. Ning. Containing denial-of-service attacks in broadcast authentication in sensor networks. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 71–79, New York, NY, USA, 2007. ACM.

[24] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, 2002.

[25] S. Zhu, S. Xu, S. Setia, and S. Jajodia. LHAP: A lightweight hop-by-hop authentication protocol for ad-hoc networks. In *Proceedings of the Workshop on Mobile and Wireless Network (MWN)*, 2003.