A Practical Study of Transitory Master Key Establishment For Wireless Sensor Networks

Jing Deng, Carl Hartung, Richard Han, Shivakant Mishra
Computer Science Department
University of Colorado at Boulder
Boulder, CO, 80309-0430

Abstract

Establishing secure links between pairs of directly connected sensor nodes is an important primitive for building secure wireless sensor networks. This paper systematically identifies two important security requirements of pairwise key setup in wireless sensor networks, namely opaqueness *and* inoculation*. Transitory master key schemes, such as the LEAP protocol, can satisfy both requirements if the master key has not been compromised. However, if the master key is compromised, every key in the network is exposed to an adversary. To prevent the master key from becoming a single point failure of the whole system, we propose a new* opaque *transitory master key (OTMK) scheme for pairwise key setup in sensor networks. In OTMK, even if the master key is compromised, an adversary can only exploit a small number of keys nearby the compromised node, while other keys in the network remain safe. To further investigate key establishment schemes, we experimented with a way to compromise a sensor node, and tested our key establishment time in a real sensor network environment.*

1 Introduction

Wireless sensor networks (WSNs) are rapidly growing in their importance and relevance to both the research community and the public at large. Security is a critical issue for many applications of sensor networks, e.g. military battlefield deployments and homeland security.

Establishing pairwise keys between pairs of neighbor nodes has become an important primitive for building sensor network security. These pairwise keys can be used to provide support for data confidentiality, integrity, and node authentication in hop-by-hop communication. These keys have also been used to defend against some sophisticated attacks such as the HELLO attack [18, 11]. Because of their importance, many schemes to set up pairwise keys in wireless sensor networks have been proposed [14, 10, 20, 13, 27, 7].

A straightforward approach to setting up pairwise keys is to pre-configure all sensor nodes with a global key and use that key as a pairwise key for all pairs of directly connected nodes, which we call neighbor nodes. However, this scheme has a serious deficiency. If the global key is compromised, an adversary can decrypt all information that is being exchanged over the network by simply sniffing the network. Another mechanism is to use the base station as the authenticated third party to assign pairwise keys for all neighbor nodes in the network. However, it is not scalable for a large multihop sensor network, in which a single base station needs to manage a large number of sensor nodes. Several sophisticated pairwise key setup schemes for WSNs have been proposed over the last several years. Examples include random key pre-distribution schemes, transitory master key based schemes, and plaintext key exchange schemes.

To analyze these key setup schemes for WSNs, let's first look at the security requirements for traditional key setup schemes. Traditional key setup introduces three security requirements: (1) confidentiality; (2) authentication; and (3) integrity [8]. However, for a large WSN, these requirements can be relaxed. As analyzed by Anderson *et.al.*, for a large sensor network the confidentiality requirement can be made less strict if an adversary is only able to eavesdrop on key setup within a small area [7]. This doesn't mean that the key setup schemes for wireless sensor networks require less security. Integrity and authenticity are still important for sensor network key setup. In addition, intrusion tolerance becomes an important characteristic required by pairwise key setup schemes in sensor networks. Sensor nodes are typically deployed *in situ* and are therefore susceptible to physical capture and compromise by an adversary. For reasons of cost, sensor nodes often lack expensive hardware protection to prevent tampering. Even with hardware protection, other problems such as hardware failure during deployment can expose information contained in a deployed node. In all these cases, compromising a sensor node allows an ad-

versary to gain access to all information stored in that node, including all of the node's keys. There are several potentially adverse consequences of a node compromise: an adversary can obtain all keys stored in the compromised node; an adversary can use the node's keys to deduce the keys employed elsewhere in the network; and an adversary can use the node's keys to aid unauthorized malicious sensor nodes to join and/or disrupt the network elsewhere.

Hence, a key setup scheme for a wireless sensor network should satisfy at least two additional properties related to intrusion tolerance: (4) the *opaqueness property* - an adversary cannot deduce most of the pairwise keys being used in the network by compromising a small number of sensor nodes; and (5) the *inoculation property* - an adversary cannot aid unauthorized sensor nodes to successfully join a network by compromising a small number of sensor nodes.

This paper first summarizes the existing pairwise key setup schemes in Section 2 and shows how they are vulnerable to compromise by being unable to satisfy one or more of the properties of opaqueness and inoculation. Section 3 demonstrates the ease and speed with which a node can be compromised using off the shelf technology, thereby exposing the vulnerability of transitory master key schemes. Section 4 describes OTMK, a pairwise key setup scheme in that improves the resiliency of the transitory master key approach so that the property of opaqueness is at least satisfied. Section 5 describes additional issues in the design of OTMK. A completed implementation and performance results of our approach are given in Section 6. Although some sensor network key setup schemes have been implemented on a sensor node [5, 26], or simulated in a dense network [14, 10, 20, 13, 7], we have not found prior work assessing the performance of these implementations under real multi-node sensor networking conditions.

2 Analysis of Current Pairwise Key Setup Schemes

A variety of pairwise key setup schemes for WSNs have been proposed over the last several years. These can be classified into the following categories: (1) Random Key Pre-distribution (RKP) schemes; (2) Transitory Master Key (TMK) based schemes; and (3) Plaintext Key Exchange (PKE) schemes. We investigate these schemes for their resiliency against node compromise, particularly in terms of inoculation and opaqueness.

2.1 Threat Model

In this paper, we assume that sensor nodes are not mobile. People deploy sensor nodes in a certain area and these nodes form a wireless ad hoc network. After that, a sensor node cannot move to other places. Because of limited power support, we believe that this assumption is suitable for most current sensor network applications.

We assume that the adversary is able to compromise a sensor node and obtain all of its keys. We show that this is a feasible assumption in Section 3. The adversary takes a certain amount of time to compromise a node T_{min}. The adversary can consist of multiple agents, each working to compromise a different sensor node, but each agent can only be in one place at one time. The number of agents is small relative to the size of the sensor network. The adversary's agents can move to another place at a later time, introducing a compromised node elsewhere in the network. The adversary could clone nodes, and introduce them in several different areas. The adversary is also able to eavesdrop on message exchanges due to the broadcast nature of the wireless medium. However, recording of such exchanges is confined to the limited number of small areas inhabited by the adversary's agents. The adversary is not able to eavesdrop on the entire network.

2.2 Random Key Pre-distribution Schemes

In RKP schemes, every node is preconfigured with a number of keys randomly selected from a large key pool. Neighboring nodes use their preconfigured keys to set up their pairwise keys. A communication channel secured between two nodes using pairwise keys is called a key path. To protect confidentiality, every key is usually assigned an index, and nodes exchange the index of keys with neighbors to ultimately determine their shared pairwise keys. If the network density, the size of the key pool, and the number of keys pre-configured in each sensor node are carefully chosen, it is highly likely that all nodes in the network will be connected via key paths.

Compromise of a node reveals its keys and any local pairwise keys, but the WSN still maintains opaqueness against an adversary. Even if a small number of sensor nodes are compromised, the majority of pairwise keys in the network cannot be deduced by the adversary, since the adversary only has access to a subset of the pool keys. However, an adversary who obtains a compromised node's keys can still *inject* malicious sensor nodes elsewhere into the network, since the pool keys that were obtained are always valid and are used to authenticate each node. As a result, RKP is unable to *inoculate* the sensor network against arbitrary malicious injections. Another problem is RKP schemes usually consume more memory than TMK schemes and PKE schemes, and memory is a very constrained resource for a sensor node.

2.3 Plaintext Key Exchange Scheme

In the PKE scheme proposed by Anderson *et al.*, every node sets up pairwise keys with each of its neighbor nodes by sending plaintext [7]. Opaqueness is achieved because an adversary cannot be in all places at all times, and therefore cannot observe the setup of all pairwise keys. If the time for this key exchange phase is short, an adversary has very little time to eavesdrop on key setup. At the most, an adversary can observe key setup in only a small part of the network. Compromising a node does not afford the adversary any added advantage in deducing these keys. PKE does not provide protection for confidentiality, integrity, and node authentication. Another drawback of this approach is that an adversary can inject malicious nodes into the network, since there is no authentication mechanism to verify whether a sensor node is a valid member.

2.4 Transitory Master Key Schemes

In TMK, the same transitory master key is preconfigured into each sensor node. A node uses this key to generate pairwise keys to share with each of its neighbors. After the key setup phase, each node erases the master key from its memory.

A representative TMK scheme is the LEAP protocol, proposed by S. Zhu *et al.* [27] In LEAP, every node is preconfigured with a master key K_I (called the initial key). For a node with ID u, its individual key is $K_u = f(K_I, u)$, where f is a secure one-way function. For nodes u and v, their pairwise key is $f(K_u, v)$, if $u > v$, and is otherwise $f(K_v, u)$. By exchanging ID numbers alone, every node can set up pairwise keys with its neighbor nodes. After the key setup phase, every node v will *erase* the master key K_I from its memory, but retain its own individual key K_v. If a new node u (who has the master key K_I) wants to set up a pairwise key with an old node v, u can compute their pairwise key by $K_{u,v} = f(f(K_I, v), u)$, and v can also compute $K_{u,v}$ from its individual key K_v and the ID of u. [1]

Since the master key will be erased from memory after the key setup phase, an adversary who compromises a node u *after* that key setup phase can only capture u's individual key K_u and the pairwise keys between u and its neighbor nodes. The adversary cannot deduce other pairwise keys in the network, and he cannot inject other malicious nodes into the network, since the adversary cannot compute $f(K_I, i)$ for $i \neq u$ during node joining. The network is therefore opaque to and inoculated against the adversary after key setup has completed.

[1] While LEAP uses the terminology "master" key to denote the individual key, we consistently use the term master key throughout this paper to denote the initial key spanning all nodes.

However, the LEAP TMK scheme loses its opaqueness and inoculation against malicious nodes if an adversary is somehow ever able to obtain the master key K_I before it is erased. In this case, the adversary will not only be able to compute all previously setup pairwise keys in the network, but will also be able to calculate all future pairwise keys that may be setup. In addition, the adversary can also inject any number of malicious nodes into the network.

The likelihood of obtaining the transitory master key, either during the TMK key setup phase or in some other manner, is an important issue. One argument is that since the key setup phase is very short, it is very difficult for an adversary to compromise a node during that time period. However, there are examples that demonstrate the feasibility of such compromise. A small number of sensor nodes with hardware faults may retain the master key in flash memory without erasing it. This scenario is increased if sensor nodes are sprinkled from a passing airplane. If the master key is stored in flash memory, then an adversary can easily retrieve this key from a malfunctioning or broken node damaged from impact. As we will show, it is also possible to read the key even if it is stored in volatile RAM. In other cases, the time to deploy a sensor network may be significant. In these cases, the setup phase may need to be extended until all nodes are activated, or reach their destination. In such cases, the master key would exist for much longer than several seconds, perhaps tens of minutes or more. Although designers will try their best to enhance the physical security of a node and protect the master key, since the master key is a single point failure of the whole sensor network system, an adversary will also try its best to compromise the master key.

To mitigate these vulnerabilities in the basic LEAP TMK scheme, Zhu *et al.* proposed enhancements to LEAP by introducing time-limited initial keys [26]. In this enhanced LEAP protocol, every master key is only valid for a time slot. Every node that is deployed at time slot T is configured with the master key M_T, and a number of individual keys for all other time periods t, $t \geq T$. If M_T is compromised, the adversary can only know the pairwise keys setup within time period T. The pairwise keys setup during other time slots are safe.

However, this solution introduces its own set of issues. A critical question is how to determine the length of the time slot T. If T is long and there are many nodes setting up keys during T, this approach is not especially advantageous compared to the original LEAP protocol. By reducing T, the number of pairwise keys that would be captured if M_T is compromised is also reduced. However, shorter intervals of T increase the difficulty of practical deployment. Nodes would have to be deployed within a short time period. If they miss their deadline and their master key expires, then they will be unable to setup pairwise keys with any nodes.

key setup schemes	opaqueness	inoculation	mobility support	new node joining
RKP	yes	no	yes	yes
PKE	yes	no	yes	yes
LEAP	yes/no*	yes/no*	no	yes
OTMK (Scheme II)	yes/yes*	yes/no*	no	yes

Table 1. Properties of pairwise key establishment schemes. (* master key is compromised)

This could necessitate expensive rekeying. Also, it may not be physically possible to deploy nodes within the short time period, especially if nodes are deployed manually, or are deployed across a wide area. Tighter time synchronization is required for shorter intervals. The number of individual keys stored per node would also be higher for shorter intervals. If the lifetime of the network is sufficiently long, then the number of individual keys that need to be stored to accommodate shorter intervals could exceed the limited memory capacity of sensor nodes.

Table 1 summarizes the properties of the above three types of pairwise key establishment schemes. We also list our OTMK scheme for comparison. We can see that the transitory master key scheme, LEAP, keeps both opaqueness and inoculation properties when the master key is secure. Our OTMK scheme has an advantage over LEAP in that when the master key is compromised, OTMK still preserves opaqueness. In terms of mobility, RKP and PKE schemes will be able to establish secure links as nodes move, while TMK schemes, such as LEAP and our OTMK scheme, don't adapt well to mobile sensor networks. This is because two long-lived mobile nodes that have already removed their master keys will be unable to establish a secure link.

3 Practical Techniques For Compromising a Sensor Node

To demonstrate the viability of node compromise of the master key, we have designed and carried out a number of experiments detailing the relative ease with which current sensor nodes can be compromised using regular 'off the shelf' products. While node compromise is often discussed as a potential vulnerability in sensor networks, almost no work has been published to prove the feasibility of such attacks on today's sensor nodes.

Our experiments found that only three readily available tools are needed for complete compromise of a typical sensor node, MICA motes: a laptop (or computer), a programming board, and a debugging interface JTAG device [3]. With only the computer and programming board, attackers can use the freely available *uisp* utility to obtain copies of the executable located in flash, and any data located in the onboard EEPROM. The code is dumped off the node in the form of a Motorola SREC, which can be easily converted into assembly language using freely available AVR tools in order to be analyzed. Thorough analysis of the code can provide the attacker with routing algorithms and security mechanisms. Further, by using a JTAG interface an attacker can also obtain the node's SRAM in addition to both the flash and EEPROM data. SRAM is generally considered a safer place to store keys because of its volatile nature. However, we were able to demonstrate for example that the keys to TinySEC [17] could easily be observed by dumping the RAM.

We found that an attacker could obtain copies of all the node's memory and data within *one minute* of discovering it, given the proper level of experience with JTAG programmers and AVR tools. The majority of that time is spent dumping the flash or RAM across the communication line, e.g. serial. Only about fifteen seconds is spent to type the commands needed to invoke the proper downloading tools. If an attacker knows the precise area of memory from which to pull the keys, then the compromise time T_{min} is in the tens of seconds.

These results demonstrate the ease and speed with which an experienced attacker can compromise a sensor node. In TMK schemes, if the master key takes tens of minutes to set up, then an adversary can quickly compromise that node within a minute of finding it. Tamper-proof hardware can slow down the adversary, though cost may not allow this defense. Even with tamper-proof hardware, damaged or disabled nodes can be made to give up their secret keys using techniques similar to the ones presented here.

4 OTMK: Opaque Transitory Master Key Establishment

In this paper, we propose a solution to pairwise key setup that improves the transitory master key setup scheme of LEAP [27]. While both TMK and RKP provide opaqueness in the presence of node compromise, we selected the TMK class of approaches as the basis for our work, because RKP-based approaches suffer from the weakness that an adversary can inject any number of malicious nodes into the network by just compromising one node. In contrast, TMK and LEAP provide some inoculation against injection when the adversary hasn't obtained the master key. In our im-

proved solution of OTMK, even if the master key is compromised, the property of opaqueness is preserved. Our basic OTMK protocol in section 4.1 achieves this opaqueness using simple techniques that ease node deployment, instead of relying on the complex timing mechanisms proposed in [26] as enhancements to LEAP. We then propose a more flexible OTMK protocol in section 4.2 that allows new nodes to join after the master key has been erased, while preserving opaqueness and inoculation after the master key has been erased.

4.1 OTMK Scheme I: a basic protocol

The protocol description adheres to the following notational conventions:

$u \rightarrow v$	:	node u sends message to node v
$u \rightarrow *$	:	node u broadcasts to all neighbors
$a \| \| b$	:	message a concatenated with b
$E_k(p)$	:	encrypt message p with key k
$M_k(p)$	:	generate MAC for message p with key k

In this scheme, all sensor nodes are pre-configured with a master key M. To set up pairwise keys with their neighbors, each node u broadcasts a JOIN request message as follows:

$$u \rightarrow * \quad : \quad JOIN||E_M(ID_u||nonce)$$

Here ID_u is the identity of the sensor node u and $nonce$ is a random number. When a node v receives this message, it generates a random number $K_{v,u}$, and sends the following REPLY message to u.

$$v \rightarrow u \quad : \quad REPLY||E_M(ID_v||nonce+1||K_{v,u})$$

When node u receives this message, it decrypts this message and verifies the nonce. If verified, it records node v as its verified neighbor. The pairwise key between u and v is either $K_{v,u}$ that v generated or $K_{u,v}$ that u generates on receiving a similar request message from v. If $ID_u < ID_v$, nodes u and v use $K_{u,v}$ as their pairwise key. If $ID_u > ID_v$, then they use $K_{v,u}$ as their pairwise key. We term this above scheme as OTMK Basic I.

To reduce the chances of master key compromise, every node destroys its master key from memory after a certain time that is long enough to set up pairwise keys with its neighbors.

This basic scheme preserves opaqueness but not inoculation if the master key is compromised. The master key M is used here only to authenticate a legitimate sensor node, not to compute all pairwise keys, unlike LEAP. This has a very important consequence. If an adversary compromises the master key after a node has set up its pairwise key, the adversary cannot deduce this pairwise key. If an adversary compromises the master key before the key setup process is over, it can find out all pairwise keys that are being exchanged during the key setup phase by observing the *reply* messages. However, since an adversary can observe only a small part of the network at any point in time, it can deduce the pairwise keys of only a small part of the network. In this way, opaqueness is preserved both before and after the master key is erased. This is in contrast to LEAP, where a compromise of the master key at any time allows an adversary to determine all pairwise keys in the network. We will have more to say on the inoculation property in section 5.2.

The overhead of the basic scheme can be improved in the following manner. The basic scheme requires that each node unicast a reply to the node that initially broadcast the Join. This incurs additional energy costs and delays key setup. To address this problem, we propose the following modification that only requires one broadcast message sent by each node for key setup. When node u sets up its pairwise keys with its neighbor nodes, it broadcasts message

$$u \rightarrow * \quad : \quad JOIN||E_M(ID_u||nonce_u)$$

where ID is the ID of sensor node u, $nonce_u$ is a random number.

If node u and v receive JOIN request messages from each other, they will generate their pairwise key with the formula

$$K_{u,v} = f(ID_u||ID_v||nonce_u||nonce_v)$$

when $ID_u < ID_v$. Otherwise, when $ID_u > ID_v$, their pairwise key is

$$K_{v,u} = f(ID_v||ID_u||nonce_v||nonce_u)$$

In this way, every node only needs to broadcast one Join message, in order to set up their pairwise key. We term this abbreviated basic scheme as OTMK Simplified I. The drawback of this scheme is that a node cannot immediately ascertain whether the pairwise key has really been setup. Nodes have to verify their pairwise keys at a later time, e.g. by sending a well-known message with a MAC generated by the pairwise key.

4.2 OTMK Scheme II: Enabling new nodes to join

Our approach should support the ability to allow new nodes to join even after the master key has been destroyed. As is, the basic scheme will be unable to add new nodes once the master key is destroyed. As a result, it is imperative that we develop a new solution capable of handling joins beyond the initial deployment.

Suppose a newly joining node u wants to setup a pairwise key with an existing node v. There are three problems that we need to solve: (1) How can v who no longer has the master key authenticate u?; (2) How can u authenticate v before setting up a pairwise key with v?; and (3) How can u and v setup a pairwise key between each other?

We propose a solution that addresses these three problems and allows new legitimate nodes to join an existing sensor network, while preserving opaqueness before and after erasure of the master key as well as inoculation following erasure of the master key. First, before a node v destroys its master key, it generates a new key $K_v = MAC(M, ID_v)$. In addition, it generates a number of *ver-*

ifiers. Each *verifier*$_i$ contains a random number r_i, and y_i, where $y_i = f(M, r_i)$. Here $f()$ is a a secure one-way function. Node v stores K_v, and $verifier_0, \ldots, verifier_k$, and destroys the master key M.

A newly joining node u is pre-configured with the master key M. To authenticate u, v sends one of its random numbers r_i as a challenge to u, where $0 \leq i \leq k$. Since u is pre-configured with the master key M, it computes $z_i = f(M, r_i)$ and sends it to v. Node v can now authenticate u by verifying if $z_i = y_i$.

This verification mechanism has some security weaknesses that need to be addressed. An adversary can launch a replay attack and an r_i exhaustion attack upon it. First, an adversary can passively monitor the network traffic to collect $< r_i, y_i >$ pairs, and then use them at some later time to join the network. Second, how many verifiers does a node need to maintain? Too many verifiers would consume excessive memory, while too few verifiers allow an adversary to send many fake Join requests and force a node to use up all of its verifiers. If an exhausted node refuses to issue challenges, then all future valid Joins will be blocked. If an exhausted node repeats an old r_i, an adversary can provide the saved y_i to fraudulently verify itself.

To guard against the above attacks, u does not send y_i directly to v. Instead, v sends a nonce n_v in addition to r_i in its challenge to u, and u uses y_i to generate a message authentication code (MAC) for n_v and sends it back to v. This mechanism is self-metered and resilient to exhaustion in that only valid Joins need use up an $< r_i, y_i >$ pair.

Finally, to authenticate v, u includes a random number n_u in its initial join request and requires v to send a MAC of n_u computed using K_v.

The complete protocol for a new node u joining a network and setting up a pairwise key $K_{u,v}$ with an existing node v is shown below:

$$\begin{aligned} u \rightarrow * \quad &: \quad JOIN||n_u||ID_u \\ v \rightarrow u \quad &: \quad ID_v||E_{K_v}(r_i||n_v)||MAC_{K_v}(r_i||n_u||ID_v||n_v) \\ u \rightarrow v \quad &: \quad ID_u||E_{K_v}(K_{u,v})||MAC_{y_i}(n_v||K_{u,v}) \end{aligned}$$

A node that doesn't have the master key will be unable to join the network. Since nodes destroy their master keys soon after initial pairwise key setup, compromising a node after that time will not allow an attacker to inject new false nodes into the network. The attacker's compromised node will not be able to generate correct responses based on y_i to the random number challenges r_i. However, a legitimate new node just introduced to the network has the master key temporarily and can securely add itself to the network.

OTMK scheme II thus preserves the opaqueness and inoculation properties after the master key M is erased. If an adversary compromises the master key, Scheme II still preserves opaqueness, because pairwise keys are generated locally, and not from the master key. Scheme II also has the advantage that it is completely decentralized and not timing dependent, unlike the enhanced LEAP approach.

5 Practical Considerations

5.1 Behavior in the presence of message loss

During the key setup process, if any message is lost, the pairwise key cannot be set up. A simple solution is to retransmit key setup messages. A node re-sends broadcast messages several times during the key setup time to make sure that its neighbor nodes can receive its message. To keep consistency, every retransmitted broadcast message or unicast message uses the same random number ($nonce$). In OTMK basic scheme I, when a node receives repeated first join (broadcast) messages, it just replies with the same unicast message, because it doesn't know whether the sender received its previous reply. In OTMK's node joining scheme II, when a new node receives repeated second challenge (unicast) messages from an old node, it will reply with the same third reply (unicast) message as before. When an old node receives repeated first join (broadcast) messages from a new node and hasn't received the third message from that node, it will reply with the same second challenge message to that node.

Naturally, an adversary can launch replay attacks against these retransmission schemes. The advantages of the replay attack are that it generates denial of service and/or energy exhaustion against the nodes nearby the adversary. These nodes have to resend many messages. Notice that these attacks only affect nodes within a small area. If we can set some threshold to limit the total number of key setup messages sent by a node within certain time, we can mitigate these attacks. This threshold can also defend against an adversary sending many invalid key setup requests. An adversary cannot capture keys or inject false keys by launching a replay attack. Also, it is hard for the adversary to launch other attacks by replaying the key setup messages in other parts of the network, since in transitory master key schemes the key setup time is limited.

The actual parameters, such as how many times to retransmit broadcast messages and how many times to retransmit unicast messages, depend on the particular implementation scenario, e.g., the network density or the number of secure links that a node desires.

5.2 Mitigating compromise of the master key

One problem with the schemes proposed in Section 4 is that if the master key M is compromised during the key setup phase, then the inoculation property cannot be achieved. An adversary can inject any number of malicious nodes into the network, although the adversary still cannot

deduce most of the pairwise keys already setup in the network. For TMK schemes, it is difficult to prevent an adversary from injecting malicious nodes if the master key is compromised, since the master key is used to authenticate a node.

There are several ways to counter false injection if the master key is compromised. First, an authenticated third party, e.g., the base station, can be used to verify every new node added in the network. This approach incurs much data traffic and energy consumption for a large scale sensor network, but is still practical for a small network. Second, the scope of the master key can be confined either by its geographic location or its validity in time. Several random key distribution schemes propose to distribute keys corresponding to a node's location [21, 13, 15]. If the keys in a node correspond to the node's geographic location, then an adversary will have difficulty cloning that node to other locations. However, one problem of geographic location based random key distribution scheme is the difficulty of deployment, since for each node, we need to know its approximate location and pre-configure a certain set of keys in it.

In the time limitation approach, the master key is bounded to a limited time period. After that period, the master key is invalidated and the node that contains that identification information cannot setup pairwise keys with valid nodes. We focus our discussion on new nodes joining the network, since they have different identification information than old nodes which are already in the network. The authentication of new nodes should be processed within that limited time period, otherwise, a new node cannot be authenticated by an old node. Here, we propose a method for a time-limited master key approach. We apply a mechanism which has the same principle as the μTESLA protocol [22], and employ a one-way hash chain as the master key in each time period to authenticate new nodes. Compared to [26], this scheme consumes less memory. Old nodes don't need to store each key for each time slot.

Every node is preconfigured with a master key H_1. That key is on a one-way hash chain $< H_k, H_{k-1}, \ldots, H_1, H_0 >$. Every master key H_i corresponds to a time slot i. For the nodes that are deployed in the same time frame, they set up keys with each other by using the master key, as described in the previous section. After the pairwise keys are established, every node v in time slot i computes K_v with H_i, and computes H_{i-1} from H_i through the one-way function, then it stores K_v and H_{i-1} and erases H_i. When a node has stored H_{i-1} and erased H_i, even it is compromised, adversary still cannot use its keys to inject malicious node.

To set up the pairwise key between a new node u and an old node v, we use the following scheme (suppose u uses H_j and v uses H_i, where $j > i$):

$$
\begin{array}{lcl}
u \rightarrow * & : & JOIN||n_u||ID_u \\
v \rightarrow u & : & n_v||ID_v||||i||MAC_{K_v}(n_u||n_v||ID_v) \\
u \rightarrow v & : & ID_u||E_{K_v}(K_{u,v})||MAC_{H_j}(n_v||ID_u||K_{u,v}) \\
u \rightarrow * & : & H_j
\end{array}
$$

First, the new node u broadcasts a JOIN message with a random number n_u to its neighbors. When an old node v receives this message, it uses its individual key K_v to MAC n_u, and sends a random number n_v, its ID ID_v, and master key index i to u. u can generate H_i with H_j and i, and then it generates K_v by H_i and verify the MAC, so u can authenticate v. If v is verified, u sends the MAC for n_v, and their pairwise key $K_{u,v}$ to v. After the key setup period, u broadcasts its master key H_j to its neighbors. Then v can authenticate u, u's ID, and the pairwise key $K_{u,v}$ by verifying the MAC generated with H_j. If an adversary compromises a node in time slot i, it cannot use H_i to inject malicious nodes in later time slots.

In general, a drawback of the time limitation approach is that it requires at least loose time synchronization in the network, which increases the complexity of network deployment and management.

6 Implementation and Key Establishment Time Estimation

6.1 Goal of experiments

We implemented our OTMK key setup scheme on Berkeley motes [2]. Our assessment includes how to initialize the transitory master key, the delay overhead of encrypting/decrypting data for each message exchange, and an evaluation of the overall key setup time for each scheme.

These experiments provide us a rough estimate of the key establishment time in current wireless sensor networks. Notice that OTMK is not intended to speed up the key establishment time. Instead, OTMK makes the transitory master key scheme more resilient to master key compromise. However, the longer the time needed for key establishment, the greater the risk of master key compromise. Our experiments show that the time required for message exchanges dominates key establishment time. Since in the LEAP protocol every node broadcasts one message for key setup, its time should be similar to our simplified scheme I. For the random key schemes such as [14, 12, 20], the number of exchanged messages are based on the key pool and the number of keys in each node [9]. R. Di Pietro *et.al.* proposed an efficient algorithm which significantly reduces message exchanges in Gligor schemes [23]. Its key setup time is similar to LEAP and OTMK simplified scheme. the setup time of RKP schemes should be no less than transitory master key schemes.

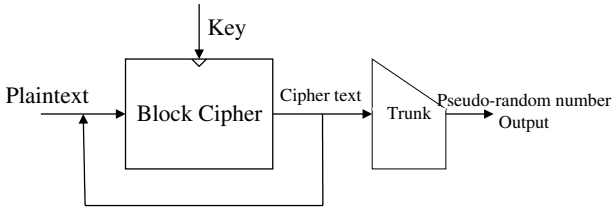

Figure 1. Secure Pseudo-random Number Generator.

6.2 Pseudo-random number generation

The security of our key setup schemes depends upon secure random number generation on a node. The random number generator (PNG) should be statistically uniformly distributed. In scheme I, since the random numbers are encrypted with master key E_M, we don't have to use a secure pseudo-random number generator. For new node joining scheme II, since a new node u needs to send a random number n_u in plaintext, if an adversary can force u to send many joining requests and eavesdrop them, e.g., jamming the reply messages from old nodes, he can crack the non-secure pseudo-random number generator and anticipate following random numbers, even the keys of u, without compromising u.

A simple solution for this problem is to use a secure, unpredictable pseudo random number generator. Current real random number generator are too heavy for sensor node platform [1, 19]. Here, we use a standard hash chain generator, which is the lightweight, and secure unpredictable random number generation scheme, for our key setup schemes. First, we preconfigure two random numbers on each sensor node; one is a *seed*, and the other is the first random number r_i. These two random numbers are generated by powerful computing equipment, such as a desktop, with a secure random number generator. As shown in Figure 1, every sensor node uses a block cipher as a secure PRNG, and generates other random numbers by the formula:

$$r_{i+1} = E_{seed}(r_i)$$

for our PRNG. Here, we depend on the block cipher's property that

1. the ciphertext is statistically uniformly distributed,
2. however many plaintext/ciphertext (r_i/r_{i+1}) pairs an adversary knows, it still cannot figure out what the key is,
3. even knowing plaintext r_i, the adversary cannot figure out the ciphertext.

Write a 8 bytes key to EEPROM	$72.5ms$
Read a 8 bytes key from EEPROM	$< 1\ ms$
Write a 8 bytes key to flash	$19ms$
Read a 8 bytes key from flash	$1ms$

Table 2. read/write key in internal EEPROM

schemes	messages	encryption	decryption
BASIC I	JOIN	0.685	0.42
	REPLY	1.01	0.82
SIMPL. I	JOIN	0.685	0.42
JOIN II	JOIN	0.685	0.42
	CHALLENGE	1.315	1.045
	REPLY	1.675	1.055

Table 3. Delay overhead of packet encryption/decryption for OTMK schemes (unit: msec).

This hash chain has the property of creating statistically uniformly distributed and unpredictable random numbers. Of course, an adversary can still compromise a node and obtain its random *seed* and predict all of its future numbers. However, since the adversary already knows all keys in that node, then cracking the random number generator won't give the adversary more information.

6.3 Master key initialization

In the transitory master key scheme, the master key is first saved in non-volatile memory, e.g., the flash or EEPROM, and is then copied to volatile memory, such as SRAM. The key in non-volatile memory is then erased. At this point, the node can begin its key setup process.

In our implementation, we measured the key transfer time of the master key from both internal EEPROM and flash memory. When a node initializes, the master key is read from a fixed address in EEPROM/flash, is copied to SRAM, and a number is rewritten to that address in EEPROM/flash. This moves the master key from non-volatile memory to volatile memory. Table 2 shows the time of reading/writing an eight byte key from/to internal EEPROM, and reading/write an eight byte key from/to flash. Notice that we have to write a whole page of data in flash, which is 256 bytes, since flash writes are on a page granularity. Eight byte keys are deemed acceptable for sensor networks [17].

6.4 Delay overhead of packet encryption/decryption

We implemented three versions of OTMK on MICA2 motes: the basic scheme I; the simplified scheme I; and the

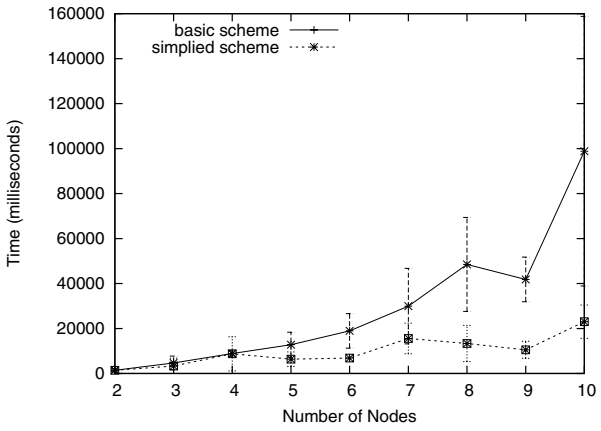

Figure 2. Total time for key setup.

new node joining scheme II. The random number length was set at four bytes long. A key is eight bytes, and a message authentication code (MAC) is four bytes [17]. Scheme II's verifier is eight bytes. To save memory, we implemented the block cipher RC5 for all encryption/decryption, MAC generation, and random number generation. We adapted the RC5 implementation from TinySEC. The packet sizes of exchanged messages are from eight bytes to twenty bytes.

Table 3 shows the time for encryption/decryption of each message in each of our key setup schemes. We see that a node spends very little time for message encryption/decryption. For example, for the second message of basic scheme I, a node only spends 1.01 ms to encrypt the message, and spends 0.82 ms to decrypt the message.

6.5 Overhead of key setup process

For the transitory master key schemes, it is important to estimate the total time for pairwise key setup in a WSN. In our tests, we used two to ten motes to execute our key setup schemes. Every node can communicate with every other node. This simulates the different densities of sensor networks, from each node having only one neighbor to each node having nine neighbors.

Our initial experiment just implemented the original protocols in section 4. In this implementation, every node tries to set up keys with its neighbor nodes as soon as possible, since the key setup time is critical for transitory master key scheme. However, because every node sends messages during a short time period, many packets collide and were lost. This resulted in failure of the key setup process.

In a later implementation, we allowed every node to execute the key setup process multiple times, in order to tolerate packet loss. As analyzed in section 5, the repeated key exchange messages don't provide a chance for an adversary to capture the key or inject false keys. An adversary can only launch limited DoS attacks on nearby nodes.

Our experiments employed the multithreaded Mantis OS (MOS) as the sensor operating system [6, 4], executing on MICA2 motes. The media access control layer protocol is BMAC [25]. We tested the key setup time for the three key setup protocols: basic scheme I, simplfied scheme I, and the new node joining scheme II.

Figure 2 shows the total time for our key setup schemes. First, we see that as the network becomes denser, the time for key setup grows longer. For example, for a two node network, the time for key setup is about 1.4 seconds, and for a ten node network, the time is about 99 seconds. As expected, the more neighbors a node has, the larger the possibility of packet collisions. Another observation is that the simplified scheme I costs less setup time compared with the basic scheme I, and the difference becomes more significant as the network becomes denser. For example, when there are ten nodes in the network, the time for completing the simplified scheme is about 23 seconds. If a node has $n-1$ neighbors, the simplified scheme I employs $O(n)$ messages, while the basic scheme I employs $O(n^2)$ messages. This experiment confirms that the number of messages for the key setup scheme will significantly affect the completion time for that protocol as the density increases.

To provide more details, Table 4 shows the average delay and median delay for setting up one pairwise key during the basic and simplified key setup processes. This table confirms two observations from figure 2: 1) a denser network requires more time for key setup; 2) the simplified scheme I uses significantly less time. From this table, we also see that the median value is less than the mean value. For example, in a ten node network, half of the keys are setup within six seconds for the basic scheme. This implies that while many keys are setup during a short time, a few keys need to take much longer to be setup. The large variance shown in figure 2 also confirms this behavior. Thus, for transitory master schemes, we can set a timeline to guarantee that most pairwise keys have been setup and thereby minimize the opportunity of master key compromise.

We also implemented the new node joining scheme II. We tested the time for a new node to setup its keys with one to five old nodes. Table 5 demonstrates these results.

These experiments reveal the latency, variance, and sensitivity to density and collisions of our schemes in a truly deployed sensor network. Due to wireless channel competition and packet loss, the time for key setup was longer than we expected. This problem is critical for transitory master key schemes since it gives more time for adversary to compromise the master key. In addition, in our experiment, we only isolated a small part of the nodes from the whole network. Due to hidden terminal problems, the key setup time in a large dense sensor network may be much worse

nodes	Basic I		Simplified I	
	Mean	Median	Mean	Median
2	1094	1093	904	1037
3	1534	1105	1368	1041
4	3266	2169	1765	1191
5	3314	2184	1486	1079
6	3920	3290	1694	1117
7	6757	3028	1985	1035
8	7184	3608	2130	1400
9	6706	3722	1981	1206
10	8897	5282	2921	1717

Table 4. Average and median time for one pairwise key setup time (unit: msec).

nodes	total time	mean time	median time
1	2468	2468	1087
2	2178	1125	1131
3	5629	2965	3198
4	3550	2342	2171
5	8411	3772	3250

Table 5. Key setup time for the new node joining scheme II (unit: msec).

than our results. In addition, our experiments only count the time for message exchanges. Some deployment issues may delay the key setup process for tens of minutes. While the key setup process can be expedited, these delays as well as the rapidity of node compromise suggest that there is still a need to provide intrusion tolerance against compromised keys in pairwise key setup.

7 Related Work

Pair-wise key setup is an important building block for sensor networks and as a result has been extensively studied in recent years [17, 14, 10, 20, 13, 27, 7, 16, 24]. These approaches were summarized in Section 2.

Deng *et. al* proposed a transitory master key based pairwise key set up scheme for bidirectional verification and neighborhood authentication [11]. The master key is only used to encrypt key exchange messages, so that even if the master key is compromised, an adversary cannot know other keys in the network. The problem of this scheme is that it doesn't provide a secure authentication mechanism to verify new nodes.

8 Conclusion

This paper's first contribution identified opaqueness and inoculation as two important properties desired by pairwise key setup schemes in WSNs, given node compromise. Our second contribution was to analyze existing pairwise key setup schemes and show how they are vulnerable to compromise by being unable to satisfy opaqueness and/or inoculation. Our third contribution demonstrated the practical ease and speed with which a node can be compromised using off the shelf technology. Our fourth contribution presented OTMK, a pairwise key setup scheme that improves the opaqueness of the transitory master key approach. Our key setup schemes have been implemented, tested, and their setup times have been evaluated on actual sensor nodes.

References

[1] Ansi x9.17. American National Standard - Financial institution key management (wholesale).

[2] Crossbow website. http://www.xbow.com.

[3] JTAG. http://www.atmel.com/.

[4] mantis website. http://mantis.cs.colorado.edu.

[5] TinyKeyman. http://discovery.csc.ncsu.edu/software/TinyKeyMan/.

[6] H. Abrach, S. Bhatti, J. Carlson, H. Dui, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han. Mantis: System support for multimodal networks of in-situ sensors. In *(WSNA'03)*, San Diego, CA, USA, September 2003.

[7] R. Anderson, H. Chan, and A. Perrig. Key infection: Smart trust for smart dust. In *12th IEEE International Conference on Network Protocols*, Berlin, Germany, October 2004.

[8] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.

[9] H. Chan and A. Perrig. PIKE: Peer intermediaries for key establishment in sensor networks. In *Proceedings of IEEE Infocom*, Mar. 2005.

[10] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, May 2003.

[11] J. Deng, R. Han, and S. Mishra. Inrusion tolerance and anti-traffic analysis strategies in wireless sensor networks. In *IEEE 2004 International Conference on Dependable Systems and Networks (DSN'04)*, Florence, Italy, June 2004.

[12] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *10th ACM Conference on Computer and Communications Security (CCS'03)*, Washington D.C, USA, October 2003.

[13] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *IEEE 23rd International Conference on Computer Communication*, Hong Kong, China, March 2004.

[14] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *Conference on Computer and Communications Security, (CCS'02)*, Washington DC, USA, November 2002.

[15] D. Huang, M. Mehta, D. Medhi, and L. Harn. Location-aware key management scheme for wireless sensor networks. In *2004 ACM Workshop on Security of Ad Hoc and Sensor Networks*, Washington, DC, USA, October 2004.

[16] J. Hwang and Y. Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *2004 ACM Workshop on Security of Ad Hoc and Sensor Networks*, Washington, DC, USA, October 2004.

[17] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, Baltimore, MD, USA, November 2004.

[18] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad Hoc Networks*, 1(2-3), September 2003.

[19] J. Kelsey, B. Schneier, and N. Ferguson. Yarrow-60: Notes on the design and analysis of the yarrow cryptographic pseudorandom number generator. In *6th Annual International Workshop on Selected Areas in Cryptography (SAC'99)*, Kingston, Ontario, Canada, August 1999.

[20] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS'03*, Washingon D.C, USA, October 2003.

[21] D. Liu and P. Ning. Location-based pairwise key establishment for static sensor networks. In *1st ACM Workshop on Security of Ad Hoc and Sensor Networks, in association with the 10th ACM Conference on Computer and Communications Security*, Fairfax, VA, USA, October 2003.

[22] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. Spins: Security protocols for sensor networks. *Wireless Networks Journal(WINET)*, 8(5):521–534, September 2002.

[23] R. D. Pietro, L. Mancini, and A. Mei. Efficient and resilient key discovery based on pseudo-random key pre-deployment. In *18th International Parallel and Distributed Processing Symposium (IPDPS'04)*, page 217b, Santa Fe, New Mexico, USA, April 2004.

[24] R. D. Pietro, L. V. Mancini, A. Mei, A. Panconesi, and J. Radhakrishnan. Connectivity properties of secure wireless sensor networks. In *2004 ACM Workshop on Security of Ad Hoc and Sensor Networks*, Washington, DC, USA, October 2004.

[25] J. POlastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *2nd International Conference on Embedded Networked Sensor Systems*, Baltimore, MD, USA, November 2004.

[26] S. Zhu, S. Setia, and S. Jajodia. Leap: Efficient security mechanisms for large-scale distributed sensor networks. http://mason.gmu.edu/ szhu1/leap.pdf.

[27] S. Zhu, S. Setia, and S. Jajodia. Leap: Efficient security mechanisms for large-scale distributed sensor networks. In *10th ACM Conference on Computer and Communications Security*, Washington D.C, USA, October 2003.