Secure Mechanism for Wireless Sensor Networks Using Keylock Matching
R. Samantha¹ Mr.A.Justine Jerold²
¹M.E (Final Year), ²M. Tech.
1,2RatnaVel Subramaniam College of Engineering & Technology, Dindigul, India

Abstract— Ensuring the security of communication and access control in Wireless Sensor Networks (shortly WSNs) is of paramount importance. In this paper, I present a security mechanism, MoteSec Aware, built on the network layer for WSNs with focus on secure network protocol and data access control. In the secure network protocol of MoteSec Aware, a Virtual Counter Manager (shortly VCM) with a synchronized incremental counter is presented to detect the replay and jamming attacks based on the symmetric key cryptography using AES in OCB mode. For access control, I investigate the Key-Lock Matching (Shortly KLM) method to prevent unauthorized access. I implement MoteSec-Aware for the TelosB prototype sensor platform running Tiny OS version 1.1.15, and conduct field experiments and TOSSIM-based simulations to evaluate the performance of MoteSec Aware. The results demonstrate that MoteSec Aware consumes much less energy, yet achieves higher security than several state of the art methods.

Keywords: Sensor networks, security.

I. INTRODUCTION

A. Related Work

A Wireless sensor network (WSN) is usually composed of several resource-limited sensor nodes, which can work collaboratively and deliver useful information to users upon queries and events. Since sensor nodes may collect sensitive information, security and privacy become a concern that cannot be ignored. Moreover, several real-world scenarios, including community and environment monitoring, smart home, need data transmitted over the network and data stored in nodes’ memories. Due to the resource-limited sensor nodes, traditional network security mechanisms are not suitable for WSNs. Inspired by the above challenges; I study the issues of secure network protocol and data access control in WSNs in order to avoid data leaking to the adversary or unauthorized party.

SPINS, on the other hand, achieves low energy consumption by keeping a consistent counter (IV) between the sender and receiver, such that an initialization vector (IV) is not required to be appended to each packet. MiniSec achieves low energy consumption by appending a few bits of the IV to each packet. Packet loss, however, would cause SPINS and MiniSec to incur more energy consumption for communication and computation, respectively. Other prior works, such as ContikiSec and FlexiSec all focus on secure network protocol and do not consider the security of data stored in nodes. Kun et al.’s method is composed of three stages: network admission control, network access control, and network access maintenance. The three stages aim to add new eligible nodes into the system, guarantee that all traffic in the system is authenticated, as well as revoke compromised nodes and update group key. However, Kun et al.’s method is unable to defend or detect replay and jamming attacks.

B. Method and Contributions

I propose MoteSec-Aware, a secure network-layer protocol for wireless sensor networks. It not only works with low energy consumption but also establishes a practical high security mechanism on TelosB motes, which run the TinyOS operating system. In fact, MoteSec-Aware provides a secure network protocol to permit data transmitted in an encrypted format in the air and a filtering capability to permit or deny data access based upon a set of rules, which are frequently used to protect the data from unauthorized access while permitting legitimate communications to pass. More specifically, I base our design on the existing security primitive, AES, which has been proven to be the most suitable block cipher for the WSNs under consideration. I denote the process of executing CFA in the AES with Offset Codebook Mode (shortly OCB) mode as AES OCFCA. In this paper, AES-OCFCA is the approach proposed to achieve the goal of secure network protocol. On the other hand, Memory Data Access Control Policy (shortly MDACP) is presented to achieve the goal of data access control. To defend against unauthorized users in accessing data, I investigate the Key Lock Matching (shortly KLM) method to define access rights in each node because of its characteristic in needing low computation. For each file, there are some corresponding locks, which can be extracted from prime factorization. Through simple computations on the basis of keys and locks, protected memory data can be accessed. Our main contributions are summarized as follows:

We propose and implement MoteSec-Aware, which is the sensor network security system built on the network layer that focuses on data access control and secure network protocols simultaneously.

Fig. 1: MoteSec-Aware network topology
MoteSec- Aware achieves lower energy consumption during communication and satisfies a high level of security without appending any additional information (e.g., initialization vector) into packets.

MoteSec- Aware is evaluated in terms of the network communication overhead, energy cost, and etc. via experiments and simulations. The results demonstrate that MoteSec-Aware is feasible and efficient.

II. SYSTEM MODEL

In this section, we first introduce the wireless sensor network topology that is adopted in our method. Then, the attack model is described.

A. Network Topology for Multilevel Access

Their relationship is illustrated in Fig. 1. There are three types of nodes, including leader node (shortly LN), function node (shortly FN), and sensor node (shortly SN), in our sensor network topology. They are classified according to their hardware resources. The network region is partitioned into physical clusters, each of which contains a FN in charge of SNs in that cluster. Depending on concrete applications, clusters may overlap such that SNs in the overlapping region are affiliated with multiple FNs.

B. Attack Model

The adversary may launch both external and internal attacks. In external attacks, the adversary does not control any valid nodes in the network. Instead, the adversary may attempt to eavesdrop for sensitive information, inject forged messages, replay previously intercepted messages, and impersonate valid sensor nodes. Moreover, we assume that the adversary can jam the communication between two nodes by transmitting signals that disrupt packet reception at the receiver. The adversary may also launch DoS attacks by, for example, false data injection or path based DoS (shortly PDoS) to deplete the energy of FNs. As for internal attacks, we do not consider that the FN will be captured. Instead, we consider that the adversary may attempt to read the data stored in FNs’ memories by, for instance, utilizing an unauthorized node to read important data from FNs arbitrarily.

In view of these vulnerabilities, the critical security requirements that need to be satisfied are summarized as follows.

1) Data Confidentiality:
   This is the basic property of a secure communication protocol in that data should be kept secret from unauthorized reading.

2) Replay & Jamming Detection:
   Communication data should be ensured to be recent and verified that an adversary does not replay or jam data.

3) Data Authentication:
   It is necessary to prevent an adversary from spoofing packets. In general, a Message Authentication Code (MAC) is used for each packet to verify whether it indeed originates from another legitimate node or is altered during transmission.

4) DoS-Resilience:
   The DoS attacks, aiming to deplete energies, must be resisted in particular for resource limited sensor nodes.

5) Data Access:
   The adversary should be detected and prevented from accessing data stored on nodes.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SPINS [1]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Counter resynchronization</td>
<td>Yes</td>
</tr>
<tr>
<td>TinySec [2]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Zigbee [3]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>With 64-bit IV</td>
<td>Yes</td>
</tr>
<tr>
<td>MoteSec [4]</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Few bits of the IV</td>
</tr>
<tr>
<td>MoteSec-Aware (our method)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1: Comparison of our four method with some state of art methods (N: Number of Nodes; ϕ: Package Loss Rate; I: Bytes of the IV)

Fig. 2: MoteSec-Aware protocol stack

III. PROPOSED METHOD: MOTESEC-AWARE

The structure of MoteSec-Aware is shown in Fig. 2. The lowest two layers are the raw hardware and hardware abstraction layers. These two layers provide all the basic services and components of the limited available resources while TinyOS resides on top of them. As Fig. 2 indicates, MoteSec-Aware is constructed within the TinyOS layer. MoteSec-Aware provides several materials, including Memory Data Access Control Policy (shortly MDACP), Event Handler, VCM, Query Logic, and Key Pool. Currently, we have implemented MoteSec-Aware on the TelosB platform. We use a symmetric key cryptosystem to encrypt the data for the purpose of data confidentiality. In what follows, we describe the proposed AES-OCFA and MDACP strategies for providing protection against outside network messages and inside memory data leakage, respectively.

A. Security over DoS through CFA with AES in OCB Mode

The DoS attacks can exhaust limited energies of FNs and possibly black out a section of the monitored area. In order to deal with DoS attacks, authentication is a necessary security mechanism for preventing the communications in the network from DoS attacks. There have been many
authentication schemes proposed for wireless sensor networks.

B. Replay and Jamming Detection with Synchronized Incremental Counter Approach

MoteSec-Aware uses a synchronized incremental counter as an IV for achieving semantic security. Specifically, the IV associated with a buffer filter is used to detect replay and jamming attacks instead of appending IVs into packets transmitted in the air. With the synchronized incremental counter, we construct VCM within each node for initializing the counter and maintaining counter synchronization between the sender and receiver. The synchronized incremental counter in each node increases one count per average delay automatically. The average delay is an experimental value, which will be shown in Section IV for our system. Also, we define the maximum counter synchronization error (MCSE) to be an experiment based delay counter, $\delta$, between any pair of nodes. In other words, when the packet transmission time is much longer than $\delta$, the jamming attack can be detected at receiver.

1) Sender Side:
The synchronized incremental counter approach at sender side is depicted in Algorithm 2. Assuming that the sender has started to send a packet to the receiver. The sender gets a counter value used as an IV from VCM. If the radio channel is clear (Step 1), then it signals radio chipset CC2420 to send out packets (Step 2); otherwise it backs off for a random period of time (Step 4) and goes to the Step 1.

2) Receiver Side:
The receiver node will receive an incoming packet after propagation delay in the air. When receiving a packet successfully, the receiver node needs to perform two checks: 1. Determine whether the packet is a legitimate one and 2. Determine whether the packet has suffered attacks.

3) Counter Synchronization:
Initially, all nodes boot up with the same counter value. When the network runs for a period of time, the counters of nodes may lose synchronization. Recent advances in secure sensor network time synchronization enable pairwise time synchronization with error of mere $\mu$s. Transmission delay between neighboring nodes are on the order of $ms$. Thus, we launch VCM to synchronize counter value based on Secure Pairwise Synchronization (shortly SPS) protocol.

IV. Pairwise Timer Synchronization: Evaluation of Maximum Counter Synchronization Error (shortly MCSE)

We first carry out an empirical evaluation of the MCSE by measuring the non-malicious end-to-end delay, $d$, between pairs of nodes. Based on the above results, we derive and evaluate MCSE. When an adversary jams and replays a packet, the end-to-end delay will get increased at least by a complete packet transmission time at the physical layer. For example, a packet payload of 16 bytes will even take a few milliseconds to be transmitted at the physical layer while the radio speed of TelosB is 256 kbps. In order to account for the physical delay caused by the adversary, we denote the time of packet payload required to be transmitted at the physical layer as $\tau$. Since $d_{avg} + 3\sigma$ are the maximum delay, which includes physical layer delay, propagation delay, and others between the sender and receiver.

Now, we will verify the above derivations. We show that the non-malicious end-to-end delay $d$ closely follows a Gaussian distribution via experiments, from which both $d_{avg}$ and $\sigma$ can be obtained to compute $\delta$. A popular solution for evaluating non-malicious end-to-end delay is to timestamp the packets operating below the Medium Access Control layer. The feasibility of Medium Access Control layer time-stamping has already been shown on typical sensor network platforms. Here, we perform the proposed PCS protocol with the counter being replaced with the timestamp in order to measure on TelosB motes.

The resultant technique is called Pairwise Timer Synchronization (Shortly PTS). Our objective is to gauge the distribution of the non-malicious end-to-end delay using timer synchronization, so that an appropriate value $\delta$, which is required in the PCS protocol, could be calculated. The PTS protocol was conducted on a pair of motes, and the non-malicious end-to-end delay $d$ was calculated for 100 independent runs while using the maximum radio transmission power ($0 \, dBm$). The above procedure was conducted for 10 different pairs of motes in order to remove any hardware specific bias. In the end, we had 1000 independent measurements of the non-malicious end-to-end delay.

V. Analysis

In this section, the performance of the proposed MoteSec-Aware method is examined: Our analysis focuses on (A) replay and jamming detection, (B) resilience against node capture attack, (C) evaluation of the minimum time required to trigger PCS, (D) semantic security, (E) data access control, and (F) energy consumption.

A. Replay and Jamming Detection

Following Section IV and Eq. (4) with $\delta$ being set to 3, we will classify large delays (> 3) as being the consequence of a jamming attack. Nevertheless, it is worth noting that the false negative probability is as low as 0.003, which can be obtained as 100%−99.7% = 0.3%, where the true delay is with 99.7% confidence. Next, the receiver uses the buffer filter to check each packet with a tuple (SrcAddr, IV) for detecting replay attack. If (SrcAddr, IV) is, in fact, an entry of the buffer filter, the buffer filter returns FALSE and drops the packet; otherwise, the tuple (SrcAddr, IV) is added to the buffer filter. Recall that the data structure of buffer filter is an array. Therefore, checking any single element in an array takes $O(1)$ time (average case). Moreover, the detection probability that can be achieved is 1.

B. Resilience against Node Capture Attack

We consider the case where the adversary not only eavesdrops on the transmitted messages but also compromises $n$ nodes to use the security information stored in them, trying to recover the coefficients of $(x, y, z, w)$. We can know that the adversary cannot break $f(x, y, z, w)$ if only $n \leq d$ nodes are compromised [29], where $d$ denotes the degree of each variable in $f(x, y, z, w)$. When the adversary has compromised $n > d$ nodes, the complexity for it to obtain the coefficients of $f(x, y, z, w)$ is $\Omega(qd+1)$, where $q$ is a prime number. If readers are interested in the security analysis of CFA, you can refer to [28] for more details.
C. Evaluation of the Minimum Time Required to Trigger

We analyze how often the pairwise counter synchronization protocol needs to be triggered. Assume that a reference node A with clock frequency $f_A$ sends two time beacons at time $t_1$ and $t_2$. Each beacon contains the count $CA(t) = f_A \times t$ observed at the reference clock at A.

VI. EXPERIMENTAL ANALYSIS AND RESULT

The following tables shows the Experimental Analysis and Result

<table>
<thead>
<tr>
<th>Payload (Bytes)</th>
<th>Packet Overhead (Bytes)</th>
<th>Security Overhead (Bytes)</th>
<th>Total Size (Bytes)</th>
<th>Energy (mAs)</th>
<th>Increase over TinyOS (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TinyOS</td>
<td>28</td>
<td>12</td>
<td>0</td>
<td>45</td>
<td>0.03</td>
</tr>
<tr>
<td>SPINS</td>
<td>28</td>
<td>17</td>
<td>5</td>
<td>45</td>
<td>0.04</td>
</tr>
<tr>
<td>TinySec</td>
<td>28</td>
<td>20</td>
<td>8</td>
<td>48</td>
<td>0.04</td>
</tr>
<tr>
<td>MiniSec</td>
<td>28</td>
<td>15</td>
<td>3</td>
<td>43</td>
<td>0.04</td>
</tr>
<tr>
<td>MoteSec-Aware</td>
<td>28</td>
<td>12</td>
<td>0</td>
<td>40</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Table 2: Communication Overhead Comparison

VII. CONCLUSION

Our method, MoteSec-Aware, is proposed and implemented for TinyOS on the TelosB platform. MoteSec-Aware is an efficient network layer security system and is the fully implemented security mechanism that provides protection for both inside memory data and outside network message. MoteSec-Aware is able to achieve the goals of much less energy consumption and higher security than previous works. This fact, apart from flexibly providing an important advantage to deployed systems, greatly facilitates researchers in porting their applications on lower cost and higher security platforms.

REFERENCES