Software-Based Remote Code Attestation in Wireless Sensor Network

Tamer AbuHmed, Nandinbold Nyamaa, and DaeHun Nyang
Information Security Research Laboratory
The Graduate School of IT & T., INHA University
253 YongHyun Dong, Nam-Gu, Incheon, Korea
Email: {tamer, nandia2000}@seclab.inha.ac.kr, nyang@inha.ac.kr

***Abstract*—Sensor nodes are usually vulnerable to be compromised due to their unattended deployment. The low cost requirement of the sensor node precludes using an expensive tamper resistant hardware for sensor physical protection. Thus, the adversary can reprogram the compromised sensors and deviates sensor network functionality. In this paper, we propose two simple software-based remote code attestation schemes for different WSN criterion. Our schemes use different independent memory noise filling techniques called pre-deployment and post-deployment noise filling, and also different communication protocols for attestation purpose. The protocols are well-suited for wireless sensor networks, where external factors , such as channel collision, result in network delay. Hence, the success of our schemes of attestation does not depend on the accurate measurement of the execution time, which is the main drawback of previously proposed wireless sensor network attestation schemes.**

***Index Terms*—Wireless Sensor Network, Security, Software Attestation.**

I. Introduction

Security in wireless sensor network (WSN) has become an important issue since several WSN applications have been developed and deployed. The constraints on memory, computation, and communication in sensor node make security a challenging task. Moreover, deploying sensors in hostile and unattended environment leads to rigid dare of that sensors are being susceptible to various attacks, including node capturing, physical tampering, and sensor's reprogramming. Therefore, sensor node program integrity and consistency are considered a crucial demand for the correctness of the WSN functionality and security. The consequences of sensor node compromising do not only affect the WSN functionality, but also the security services of WSN (i.e., data confidentiality, integrity, and authenticity). Moreover, by sensor node compromising, the attacker becomes capable of launching several attacks as data forgery, selective packet forwarding, and denial of service (DoS). The direct solution to countermeasure these security concerns is applying a memory integrity check mechanism with efficient attestation protocol [1] such that the attacker, who captures the sensor node and tries either to modify the original program or to upload a new code, is detected with high probability by the WSN attester. To overcome these security problems, we propose an attestation protocol with sensor node memory check notion. In the attestation protocol, base station (BS) verifies node's program integrity and the whole memory contents. Attesting the whole memory prevents the attacker from using the extra memory space of sensor to save the original program. In case of node compromising, the attester can use different reaction mechanisms which are out of the paper context such as, simply broadcasts the sensor ID in WSN as compromised node, or reprograms the node. Hence, code attestation is considered as a significant technique to detect node compromising in WSN when attester or BS detects anomalous behavior of a specific sensor node.

In this paper, our contributions are summarized as follows:

- We propose two different techniques to fill the empty memory space of the sensor node with incompressible random noise filling, viz., pre-deployment noise filling and post-deployment noise filling, in order to prevent free space used by the attacker.
- We enhance the computation and security of the original block based pseudo-random memory traversal algorithm [2] by proposing a dynamic block based pseudo-random memory traversal algorithm.
- We propose two different secure protocols for attestation purpose, where any of them can be used according to the underlying WSN operability. Also, we show a simulation analysis of the computation and communication overhead of our attestation protocols and their feasibility in real sensor.

The notations used throughout this paper are given in Table I.

TABLE I
NOTATIONS.

Term	Declaration
N_i	ID of sensor node i
$E_{k_{a,b}}()$	Symmetric encryption between a and b parties
K_{BS,N_i}	Symmetric key between BS and sensor node N_i
Se_i	Seed for noise generation for sensor node N_i
S_i	Challenge sent to sensor node N_i for block-based pseudo-random memory traversal algorithm.
C_i	Computed checksum value of sensor node N_i
TS_i	Timestamp of the synchronized WSN which is tightly equal in both of the BS and WSN's nodes
t_{min}	The minimum time required to compromise the sensor node by the attacker [3], [4]
key_i	One-time key generated from hash of the previous key_{i-1} as $H(key_{i-1})$
b	Computed Block size used in the pseudo-random memory traversal algorithm

978-1-4244-4148-8/09/$25.00 ©2009

The rest of this paper is organized as follows. In Section II, we introduce an overview of the literature works related to attestation techniques. In Section III, we show the assumptions and the threat model for sensor node attestation protocol that we consider. Furthermore, Section IV covers the description of our protocol followed by further analysis that demonstrates the merits of our contributions in Section V. Finally, we draw the conclusions and introduce for further works in Section VI.

II. Related Works on Remote Code Attestation

In this section, we describe the conventional schemes and the implementations of the remote code attestation motivated our proposals. Also, we states the drawbacks of each of those conventional schemes including computation complexity, communication overhead, and time sensitivity of the attestation protocol's response.

The prior works in code attestation can be categorized into two work streams. Hardware-based and software-based approaches. All hardware approaches for the sensor node attestation are based on using the Trusted Platform Module (TPM) which is specified by Trusted Computing Group (TCG) [5] as the trust anchor (sensors) for attestation protocols [6], [7]. The anchors report the hardware and software configuration status to the remote attester such that the attester individually checks each sensor received status and decides whether nodes were being compromised or not. However, it is infeasible to install a TPM in each individual sensor node due to the cost consideration. To fulfill these constraint, Krauß *et al*. adopts TPM in hybrid sensor networks [8]. Therein, sensor nodes are organized in clusters assuming the cluster heads are non-resource constrained and equipped with TPM. This approach has several drawbacks include the requirements of cluster heads to perform heavy cryptographic operations and do not modify the memory configurations during the sensor runtime.

Unlike the previous approach, software-based attestation is considered promising for WSN due to size, cost, and power limitation of sensor nodes. In [2], Yang *et al*. introduced distributed software-based attestation which performs pseudo-random memory traversal over memory whose empty space is filled with incompressible random noise before the deployment of the sensor nodes. The advantages of this scheme are the computation efficiency of the block-based traversal, which we will mention later, than cell-based traversal [9] and the distributive manner of the attestation protocol which could be effective in the case where the trusted verifier (e.g., BS) can not enter the transmission range of the sensor node directly. However, secret share distribution, univariate polynomial evaluation for attested node, and extensive communication overhead are the main drawbacks of their scheme. Moreover, the scheme is vulnerable to that a compromised node is cheating after one attestation round. In [1], [9], [10], authors introduced another remote software-based attestation to verify the integrity of a code running on embedded device. The main idea behind their techniques of software-based attestation protocols is to check the correctness of response value which is computed pseudo-randomly by cryptographic hashing and/or checksum over memory content of the device. Meanwhile, those proposed schemes use attestation routine to hash the whole sensor memory in such a way that the routine cannot be optimized further (i.e, compressed). Therefore, the adversary can not exploit the size difference in order to inject a malicious code without detection. Also, these schemes assume direct sensor node attestation (i.e, single hop attestation). This is not always possible since trusted verifier cannot always enters the transmission range of all sensor nodes. Moreover, time measurement evaluated by attester in all existing software-based attestation schemes is sensitive to external factors such as network channel collision and network delay. Specially, when we have wireless environment where a problem of the communication latency of even a one-hop can not be determined. Thus, it is impossible to design a reliable one-hop and/or multi-hop attestation scheme without designing a complete timing-independent attestation protocol or at least to reduces time dependency into a fair level. The fear level makes it easily to distinguish between the presence of the attacker modifies sensor's memory and network latency. Nevertheless, the design of fully independent attestation scheme becomes infeasible considering resource constrained environment and the lack of cryptographic primitives in WSN. In the design of our attestation schemes, we focus in making it less sensitive to attestation protocol's response time compare with existing schemes rather than designing time independent scheme.

III. Assumptions and Threat Model

A. Assumptions

In this paper, the BS is assumed to be a secure attester who knows memory content and hardware architecture, clock speed, memory architecture, and instruction set architecture (ISA) of the sensor nodes. During initial code installation on sensor node, free space of the memory is filled with incompressible high-entropy random noise and the content of the memory is known by the BS. If noise filling is not available before the deployment of the sensor nodes due to some applications specification, the sensor node will then access the program memory and fill the memory after the deployment time. Also, a pairwise shared key between the BS and the sensor node is used to construct secure communication channel. Like all the previous schemes [9], [11]-[12], once the sensor node is compromised, the sensor becomes under the full control of the attacker without any hardware modification of the sensors such as enlarging the sensor memory, supporting virtual memory [13], [14] and increasing processor speed. Moreover, we do not assume any tamper-resistance hardware to be installed on the sensor.

B. Threat Model

Sensor network is usually assumed to operate in a harsh, unattended, and hostile environment. Also, sensor nodes lack of expensive tamper-resistant hardware due to the low-cost requirement. Thus, the adversary, who physically compromises

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

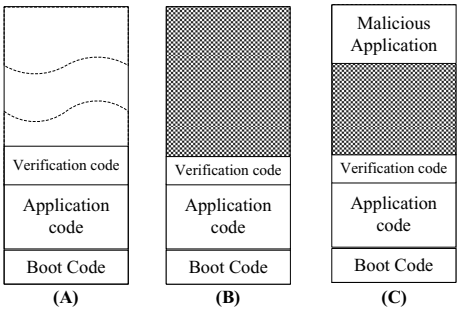


Fig. 1. Memory layout of (a) a normal sensor node, (b) a sensor node filled with incompressible random noise, and (c) a sensor node with malicious code.

the sensor nodes and uploads a malicious code, can launch various kind of insider attacks including passive attacks like eavesdropping, and active attacks like fake data injection. Furthermore, the compromised nodes can collude, by sharing their resources, to pass the attestation process as we illustrate later in security analysis section V.

IV. Proposed Attestation Protocols

A. Noise Filling Techniques

For memory filling, we focus on the program memory of the sensor node rather than data memory space since data memory space is much smaller than program memory space. That means the amount of available space for the attacker in data memory is assumed not large enough to put original code [2]. For normal case, the sensor node memory, after a specific application program are installed, is illustrated in Fig. 1(a), where a certain free space remains empty. This is always the case since sensor nodes are manufactured for various type of application and usually the program size of sensor cannot be estimated in advance. However, the free space in memory is likely to be a target for accommodating malicious code, as shown in Fig. 1(c). Depending on WSN applications, we propose two filling approaches to achieve the goal of filling the free memory space with random noise known to the attester. The memory should be, after filling, as shown in Fig. 1(b). Although filling the empty memory of sensor node with incompressible random noise by attester before the deployment is simple and effective, this is not applicable for all WSN applications. Thus, for filling the sensor's memory after the deployment, a post-deployment noise filling technique is proposed.

1) Pre-deployment Noise Filling: Pre-deployment noise filling is done by the attester and is summarized as follows:

1) For each sensor node, the trusted attester generates and stores a table whose entries are $< N_i, K_{BS,N_i}, Se_i >$, where node ID N_i, pairwised key K_{BS,N_i}, and seed Se_i used in the incompressible random noise generation. Attester keeps these information for attestation stage to emulate and evaluate sensor node memory.
2) The trusted attester loads sensor node N_i with K_{BS,N_i} and then fills the empty space of the memory with random noise generated from Se_i.

After these steps, the nodes are ready to be deployed and it contains an application code, the attestation function, and the cryptographic primitives. The memory layout of the node would be, as shown in Fig.1(b).

2) Post-deployment Noise Filling: Post-deployment noise filling technique is designed for scenario where the attester could not exactly estimate memory consumption by sensor application prior deployment. For instance, this case occurs in sensor's application which needs collecting information from the physical environment before the data processing stage. Thus, the sensor node is loaded with the application code, the attestation function, and cryptographic primitives, and then is deployed without filling the empty memory. Later, the node generates incompressible random noise by using a combination of sensor data and hash chain value. The filling steps are described as follows:

1) For each sensor node, the trusted attester generates and stores a table whose entries are $< N_i, K_{BS,N_i}, K_i >$ where N_i is node ID, K_{BS,N_i} is a pairwise key, and K_i, is a one-time hash chain key. Hash chain key is generated as $K_i = H_i(H_{i-1}(\ldots H(k_0)\ldots)$.
2) The trusted attester loads sensor node N_i with K_{BS,N_i}, K_i, and t_{min}, and consequently, the sensor node becomes ready to be deployed.
3) Within time t_{min} after the deployment, the sensor node uses either routing or sensed data D_i as seed for a block pseudo-random memory traversal (BPMT) algorithm shown in Fig. 2. By using the seed D_i and K_i, BPMT algorithm run with computation complexity $O(\frac{m\ln m}{b})$ to fill the whole memory.
4) The sensor node deletes the key_i and sends the D_i data to the attester so that later, in the attestation time, the attester can emulate and evaluate the sensor memory layout.

B. Pseudo-random Memory Traversal

Cryptographic one-way hash function is used in several existing code attestation schemes including Spineliss scheme [12] and Program Integrity Verification (PIV) scheme [11]. Whereas, in PIV scheme, a new hash algorithm is generated for each attestation round and sent to sensor node along with the attestation request. In SWATT [9], another technique called a pseudo-random memory traversal is used to compute checksum over the memory cells of sensor node. Though SWATT scheme is sensitive to the attestation response time and computation overhead, the pseudo-random memory traversal used in the scheme is immune against attacker misbehave. Yi Yang *et al*. [2] proposed a distributed software based attestation for WSN, where more efficient pseudo-random traversal is implemented, compared to SWATT. In Yi Yang *et al*. scheme, a block-based pseudo-random memory traversal is proposed to improve the cell-based traversal of SWATT in the sense that less number of traversals are required to verify the same amount of memory space. These schemes of the memory traversal approach depend on the Coupon

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

Collector's Problem [15]. The result of Coupon Collector's Problem states that the cell-based memory traversal iterates more than $O(m\ln m)$ times to access each memory cell at least once with high probability, where m is memory size. However, for the same memory space, the block-based traversal requires $O(\frac{m\ln m}{b})$ iterations, where b is block size [2], in order to obtain equal probability as in cell-based memory traversal. In our proposal, we modify the original BPMT to make it more immune from attacker misbehave. Hence, we propose two algorithms: (*i*) fixed block size BPMT, where the block size is not predetermined as the original algorithm. However, attester sets the block size in each attestation round explicitly or implicitly. This step makes traversed blocks unpredictable to attacker. Also, (*ii*) a dynamic block size BPMT algorithm is proposed. The goal of the dynamic BPMT is to enhance the computation complexity and decreasing the average access per cell comparing to the fixed BPMT, as we will see in section V-A. The fixed and dynamic BPMT algorithms are shown in Fig. 2 and 3, respectively. $RC5$ [16] with 64 bit block size and encryption key key_i is used and recommended to be used as Pseudo-Random Number Generator (PRNG) inside the BPMT. For WSN, $RC5$ is the most efficient and adequate cryptographic algorithm because of its high performance and low memory space requirements [17]. Hence, in Mica2 mote [14], $RC5$ of BPMT takes attester value S_i as a seed to produce 8-byte value treated as four memory address, i.e., memory address is 2-byte [18], namely t_0, t_1, t_2 and t_3. For each memory address, 1-byte of checksum C is updated based on the result of the bitwise XOR of b number of consecutive memory cells. Implementation of $RC5$ is already available in TinySec [19]. $RC5$ running in counter mode (CTR) is used due to the fact that each memory cell is individually accessible by encrypting the right counter [2]. Though $RC5$ is patented by RSA Security, $RC5$ could be easily replaced with other block ciphers schemes such as MISTY1, Rijndael or Skipjack [8].

C. Proposed Attestation Protocols

In this subsection, we present two secure attestation protocols. The protocols follow the conventional challenge and response protocol shown in Fig. 4(a). The attestation protocol can be executed in different scenarios. For instance, the BS schedules the attestation protocol to be periodically executed to guarantee the correctness of WSN functionality and to exclude any compromised node. Besides, the attestation protocol can be executed as a response action of the intrusion detection system alarm, when a misbehave action is detected by WSN administrator. In attestation time, the attester sends securely a random seed for pseudo-random memory traversal algorithm. Afterword, the algorithm makes a verification to the memory of the suspected sensor node by computing a checksum over the whole memory layout including application code. Finally, sensor node sends the checksum back to the BS which decides the correctness of memory content of the sensor node depending on the result of the checksum computed in pseudo-random memory traversal. The following subsections introduce the two different attestation protocols used for attestation purpose.

```
Input  : n, number of iteration
         S_i, seed
         key_i, encryption key for RC5
         Flag, decide algorithm operation
Output: C (C_0, C_1, ..., C_7), checksum value
C is initialized to key_i;
tmp is 1 byte initialized to 0;
j is initialized to point to C_0, the first byte of C;
b is given as input or computed from S_i;
for i=1 to n/4 do
    (t_0, t_1, t_2, t_3) = RC_{key_i}(S_i);
    if Memory Read then
    for k=0 to 3 do
        for r=0 to b-1 do
            tmp = tmp ⊕ Mem[t_k + r mod m];
        C_j = C_j + tmp;
        tmp = 0;
        j = (j + 1) mod 8;
    else Memory Write
    for k=0 to 3 do
        for r=0 to b-1 do
            Mem[t_k + r mod m] = Mem[t_k + r
            mod m] ⊕ t_k ⊕ r;
```

Fig. 2. Algorithmic description of the fixed BPMT algorithm.

```
Input  : n, number of iteration
         S_i, seed
         key_i, encryption key for RC5
Output: C (C_0, C_1, ..., C_7), checksum value
C is initialized to key_i;
tmp is 1 byte initialized to 0;
j is initialized to point to C_0, the first byte of C;
for i=1 to n/4 do
    (t_0, t_1, t_2, t_3) = RC_{key_i}(S_i);
    b = function of (t_0, t_1, t_2, t_3) mod m;
    for k=0 to 3 do
        for r=0 to b-1 do
            tmp = tmp ⊕ Mem[t_k + r mod m];
        C_j = C_j + tmp;
        tmp = 0;
        j = (j + 1) mod 8;
```

Fig. 3. Algorithmic description of the dynamic BPMT algorithm.

1) Attestation Protocol I: This proposed protocol is used when there in absence of time synchronization between the sensor nodes and the BS. Thus, the attested sensor node authenticates the BS attestation request using challenge response protocol before starting its attestation algorithm over its own content. By doing so, the sensor nodes are immune from

978-1-4244-4148-8/09/$25.00 ©2009

the DoS attack which could be launched by replaying the attestation request and exhausting the power of the sensor node. This attestation protocol consists of four steps, viz., attester request, node checksum computing, node response, and attester verification. The steps are described as follows.

1) **Attester request**: In this phase, the attester or BS sends attestation request to initiate the verification of memory content of targeted sensor node. The request includes three communication steps for sending a challenge value s_i which would be used in the next step by BPMT algorithm. These steps are shown in Fig. 5(a) and described as follow:

   BS $\rightarrow$ Node(N_i):

   $$Req_i, ID_{BS}, ID_{N_i}, E_{K_{BS,N_i}}(ID_{BS}, S_i)$$

   Node(N_i) $\rightarrow$ BS:

   $$ID_{BS}, ID_{N_i}, E_{K_{BS,N_i}}(ID_{BS}, S_i||r_i)$$

   BS $\rightarrow$ Node(N_i):

   $$ID_{BS}, ID_{N_i}, E_{K_{BS,N_i}}(ID_{BS}, r_i)$$

   In the protocol, a secure request is firstly initiated from the BS including a random challenge S_i to the sensor node N_i. Afterward, a sensor node N_i decrypts the request and verifies the ID_{BS}. Nonetheless, this step is not sufficient to approve the BS request freshness. Hence, the node verifies the freshness of the request by sending back another encrypted random challenge that includes $s_i||r_i$. Finally, BS decrypts the challenge and sends back an encrypted challenge including r_i which is verified by N_i. If these steps are correctly executed, the sensor node assumes this request as a valid attestation attempt and starts the next step (i.e., the attestation procedure). Otherwise, the sensor assumes this request as an illegal or replay request.

2) **Node checksum computing**: After N_i successfully authenticates the BS request, N_i computes checksum C_i over whole memory layout by using one of the BPMT algorithms with the input of iteration number n, seed S_i, and current key hash chain key_i as described and illustrated in section IV-B and in Fig. 4(b), respectively. For the memory with size m and block size b, the result of Coupon Collector's Problem [2] suggests n as order of $O(\frac{m \ln m}{b})$ to access each memory cell at least one time with high probability.

3) **Node response**: Therein, the target node N_i sends the computed checksum as response value C_i to the attester, as illustrated in Fig. 5(a). The response message of the sensor node is different in each attestation round and depends on the pair values S_i, the challenge and C_i. Node N_i encrypts the checksum as $E_{K_{BS,N_i}}(ID_{BS}, S_i||C_i)$, in order to prevent replay attack. This step is depicted as follow:

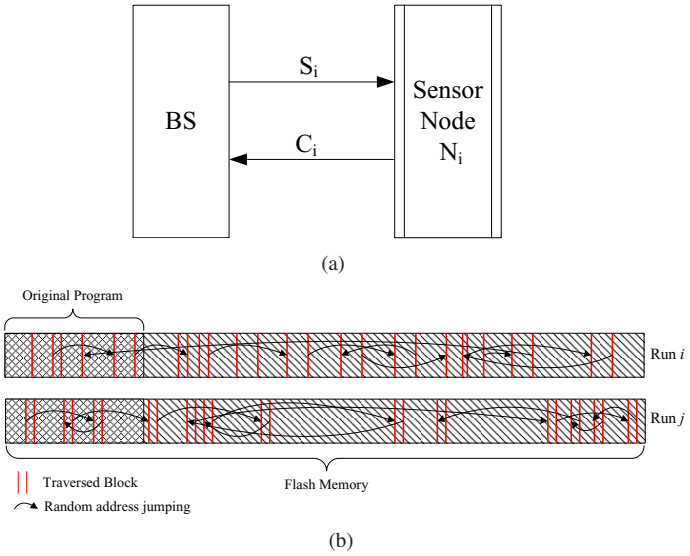


Fig. 4. (a) The main communication pattern of the attestation protocols. (b) Fixed BPMT algorithm for checksum construction C_i, where the block size of run i is different than the run j.

   Node(N_i) $\rightarrow$ BS:

   $$Res_i, ID_{BS}, E_{K_{BS,N_i}}(ID_{N_i}, S_i||C_i)$$

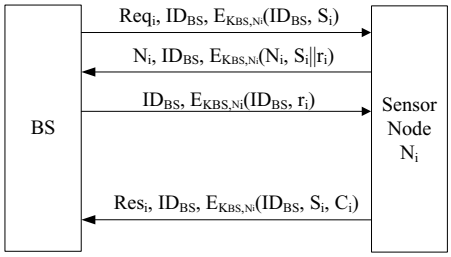
4) **Attester verification**: Finally, the attester authenticates the response by comparing ID_{BS} with the decrypted node ID_{BS} from $E_{K_{BS,N_i}}(ID_{BS}, S_i||C_i)$ and retrieves the check sum value C_i. Since attester knows the memory content of targeted node N_i, including the incompressible random noise loaded before or after the sensor node deployment, the attester internally recomputes the checksum $C_{correct}$ over N_i's emulated memory layout using the same BPMT algorithm. In the case of $C_{correct} \neq C_i$, the attester either puts sensor in blacklist by broadcast its identity as compromised node or undoes the changes of the attacker by reprogram the node.

2) Attestation Protocol II: There are many WSN applications deals with real time monitoring of the physical phenomena. For such monitoring applications, physical time often plays a crucial role. Thus, time synchronization service among sensor nodes in one side, and between BS and sensor node on the other side, is already applied and could be exploited for our attestation scheme [20]. Hence, in the presence of time synchronization service, the communication overhead of the attestation protocol is decreased comparing to the aforementioned protocol. Accordingly, our attestation protocol, for the synchronized WSN, consists of the following communication steps, and is shown in Fig. 5(b).

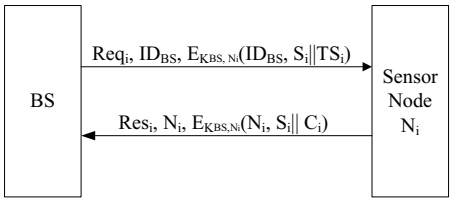
BS $\rightarrow$ Node(N_i):

$$Req_i, ID_{BS}, E_{K_{BS,N_i}}(ID_{BS}, S_i||TS_i)$$

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.



(a) Attestation protocol of an unsynchronized WSN.



(b) Attestation protocol of a synchronized WSN.

Fig. 5. Communication protocols of the software-based attestation scheme.

Node(N_i) $\rightarrow$ BS:

$$Res_i, ID_{N_i}, E_{K_{BS,N_i}}(ID_{N_i}, S_i||C_i)$$

1) **Attester request**: In this phase, BS sends an attestation request to the targeted sensor node in order to initiate memory verification. The attestation request includes an encrypted challenge S_i which is used in the next step as $E_{K_{BS,N_i}}(ID_{BS}, S_i||TS_i)$. Appending the timestamp TS_i to the encrypted request protects the sensor from the reply attack and guarantees request freshness.
2) **Node checksum computing**: The sensor N_i repeats the same step mentioned in the previous protocol for checksum computing. Therefore, After N_i successfully authenticates the BS request, it computes checksum C_i over whole memory layout by using BPMT algorithm with the input of iteration number n, seed S_i, and current key hash chain key_i as illustrated in in Fig. 4(b). Afterward, N_i encrypts the response as $E_{K_{BS,N_i}}(ID_{BS}, S_i||C_i)$ and sends it back to the BS.
3) **Attester verification**: The attester authenticates the response by comparing ID_{BS} to the decrypted ID_{BS} from $E_{K_{BS,N_i}}(ID_{BS}, S_i||TS_i)$. Then, BS computes $C_{correct}$ over N_i's emulated memory layout and checks if $C_{correct} \stackrel{?}{=} C_i$.

V. Performance and Security Analyses

A. Performance Analysis

In order to evaluate the feasibility of our attestation schemes, Java simulation has been implemented. The goal of our simulation is to evaluate the performance of our attestation schemes in term of efficiency, computation power and time. We consider the most popular sensor node platform with the following specifications: mica2 motos with 128KB data and programming memory, and 512KB flash memory [14]. TinySec [19] link layer security architecture with a variable 37 byte (i.e., 29 byte payload) is used. AS shown in Fig. 6(a), a power consumption is computed for several simulated memory traversal techniques, where the energy consumption of $RC5$ is gotten from [17]. The figure shows the energy consumption level of the cell based and the other two BPMT algorithm as the amount of the covered memory is increased. Apparently, the dynamic BPMT is the most energy efficient algorithm. For instance, $744\mu J$ is the energy consumption, when 90% of the memory is traversed by the dynamic BPMT algorithm comparing to 18mJ in case of cell based approach. Also, time evaluation of these memory traversal approaches is computed and illustrated in Fig. 6(b). Finally, Fig. 7 illustrates the average cell accessing by those memory traversal aproaches. Average access means how many times each memory cell is included in the computation of the checksum per one attestation request. For instance, to traverse 80% of the Mica2 memory, the avarage percent of cells which are traversed more than once in cell-based traversal approach is 56.52% , whereas in Fixed and dynamic block based algorithms it is $0.39\ \%$ and 0.25%, respectively. The communication overhead of our attestation protocols consist of two main parts: message transmission and message receiving. . According to [21], Mica2 has data transmission rate of 26 μs/bit, receiving current of 7.0 mA, sending current of 21.5 mA, and 3V power supply. The energy consumption becomes as follow:

- $E_{Transmission}$ = 3 $\times$ 21.5mA $\times$ (26 $\times$ 10^{-6}s/bit $\times$ 296 bits) = 0.5426 mJ/message.
- $E_{Reception}$ = 3 $\times$ 7.0mA $\times$ (26 $\times$ 10^{-6}s/bit $\times$ 296 bits) = 0.1617 mJ/message.

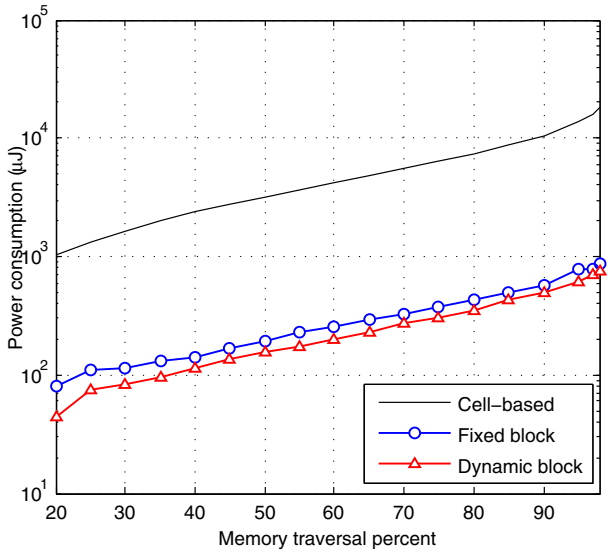
Hence, the communication overhead of protocol I is 2 received and 2 transmitted messages with total energy consumption of (1.4084 mJ). On the other hand, the communication overhead of protocol II is 1 received and 1 transmitted messages with total energy consumption of (0.7042 mJ). Regarding to the memory overhead, BS do not have to keep the entire memory of each sensor node. For instance, the pre-deployment filling technique, mentioned in subsection IV-A1, needs one record for each node includes the seed S_ie, in order to generate the incompressible memory filling. Afterwards, at the attestation time, BS generates the memory layout of the sensor node N_i on the fly to attest the node. Hence, the verifier's permanent memory of 10,000 nodes is computed as: (10,000 (nodes) $\times$ 2B (node's ID) $\times$ 8B (node's key)$\times$ 8B (Seed)) $\approx$175 KB.

B. Security Analysis

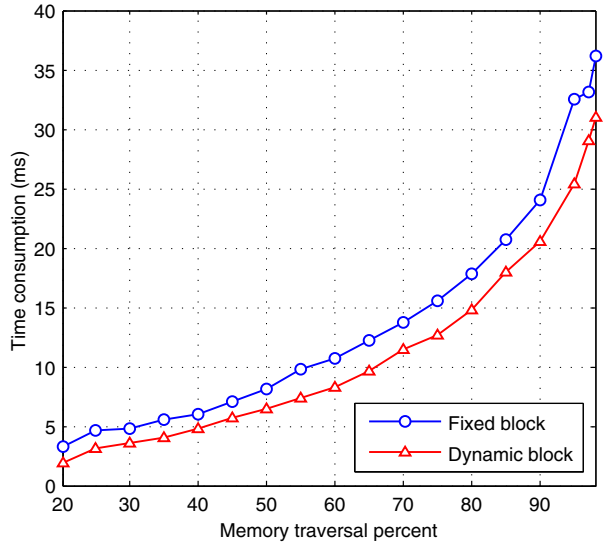
In this section, we discuss the immunity of the pre/post-deployment noise addition attestation protocols under the assumptions and the threat model given in Section III. Furthermore, we show the security properties of our protocols against possible attacks which are considered separately as follows:

Replay attack and pre-computation of checksum: In this attack, we assume the attacker compromises sensor node and tries to compute the checksum over the whole

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.



(a) The energy consumption of the the cell based and the two different BPMT algorithms.



(b) The execution time comparison of the Dynamic and static BPMT algorithms.

Fig. 6. The power and time consumption of the BPMT algorithms on Mica2 with block size 16KB.

memory and stores all possible values for the checksum before installing malicious code. Then, when attestation begins, the compromised node may attempt to respond with the pre-computed checksum to be attested properly. However, this attack is prevented by our protocols using sufficiently large initial random seed in the pseudo-random memory traversal algorithm. Moreover, since our attestation protocols are challenge-and-response with uniformly random seed, replay attack for forgery succeeds if and only if the seed is used more than once. However this situation occurs with negligible probability.

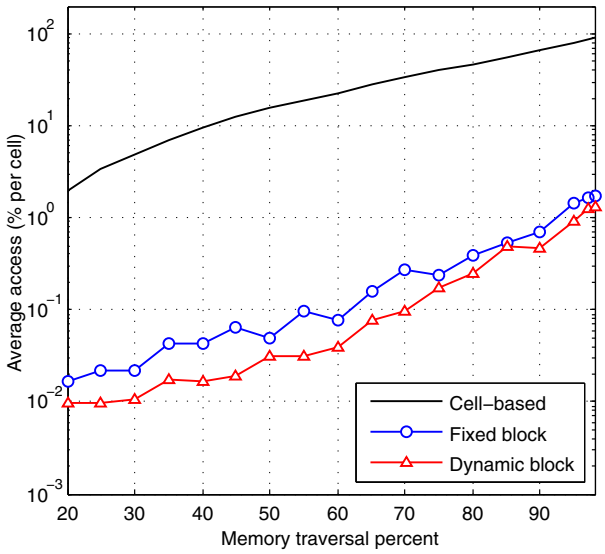


Fig. 7. The average percent of cells accessed in BPMT algorithms on Mica2.

Memory copy attack: In memory copy attack, the attacker have two ways: (*i*) copies the original sensor's code in another location of the sensor's memory and replace it with a malicious code. (*ii*) keeps the original code and copies the malicious code to any place in sensor memory. In [1], [9], this kind of attack is detected by measuring exact time of response. Thus, adoption of those schemes requires careful consideration in WSN, where response timing is sensitive to external factors such as network channel collision and network delay. This makes the one-hop latency non-deterministic in wireless environment. However, the memory copy attack is completely prevented in our attestation protocols where all free memory is filled by high-entropy noise so that no free space remains for malicious code. If malicious code is installed by overwriting the certain random values on sensor node memory, it is impossible for the adversary to compute the correct checksum, where consequently, the adversary fails to be attested properly. Also, for the post-deployment filling scenario, attacker cannot generate the random filling, since the filling algorithm needs key for $RC5$ which is a hash key chain that was removed from sensor node after filling. Thus, online memory computing is also not applicable.

Proxy attack: As described in [1], proxy attack happens if a part of memory content of a compromised sensor node is copied to another compromised node so that malicious code can be installed on that part. Later, when attestation request is received, the compromised nodes attempt to come up with correct response by computing checksum over appropriate memory content. This type of attack is applicable if traversing is sequential, as in [22]. However, our protocol computes checksum in pseudo-random traversal manner which makes next address of memory to be traversed unpredictable. Due to unpredictable next address, checksum computation requires

978-1-4244-4148-8/09/$25.00 ©2009

jumping from memory content of the first compromised node to the second compromised node or vice versa depending on where the next memory content is. However, the several jumping's in order to compute the checksum result will need communication delay between nodes and a delay in attestation's response time, which is detected easily by setting some threshold time.

Required random content on demand: The malicious code on the compromised node generates the required memory content at the time the hash function or checksum needs that memory content. In [22], the content which fills the empty memory is generated by hash function. Thus, the attack that computes the random content on demand is applicable by computing hash function several times without any need for large memory. In pre-deployment noise addition protocol, the content which fills the memory space is random noise rather than the value computed in systematic way such as hashing. Thus, filling memory with random noise prevents generation of required random content on demand. However, in post-deployment noise addition protocol, the content to fill the empty memory is generated by encrypting sensored data with one-time key as described in section IV-A2. Later, if the adversary who even knows the sensored data succeeds to compromise the sensor node, malicious code is not able to generate the memory content since key used in $RC5$ is destroyed after one time use.

VI. Conclusion and Further Work

Node attestation is considered as a crucial technique for detecting node compromising in WSN. Nevertheless, it is considered as a challenging problem due to the limited resources of the sensor node. Installing TPM in individual sensor node is infeasible because of the basic WSN requirements such as large scale and low cost. Therefore, software-based code attestation is a promising solution to detect a compromised node. However, previously proposed software-based protocols have main limitations. These limitation related to time dependency, which makes accuracy of compromised node detection questionable, and its security. In this paper, we presented two simple but practical protocols for wireless sensor node attestation. Both protocols exploit the memory filling technique to prevent the attacker from storing or changing the memory content of sensor nodes without detection. Consequently, both protocols have various advantages such as loosely time dependency compared to conventional attestation protocols, implementation feasibility, immunity to the known attacks in code attestation, and their flexibility to all type of sensor network applications. Moreover, unlike sensitive time dependent software attestation techniques, which are not applicable in scenarios that attestation along multiple hops is required, our protocols can be investigated more extensively in the future work to be extended to multi-hop node attestation. This is because our protocols do not require accurate measurements of the execution time of attestation code.

References

[1] Arvind Seshadri, Mark Luk, Adrian Perrig, Leendert van Doorn, and Pradeep K. Khosla. Scuba: Secure code update by attestation in sensor networks. In *Workshop on Wireless Security*, pages 85–94, 2006.

[2] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. Distributed software-based attestation for node compromise detection in sensor networks. In *SRDS*, pages 219–230, 2007.

[3] Jing Deng, Carl Hartung, Richard Han, and Shivakant Mishra. A practical study of transitory master key establishment forwireless sensor networks. In *SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, pages 289–302. IEEE Computer Society, 2005.

[4] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. Leap: efficient security mechanisms for large-scale distributed sensor networks. In *ACM Conference on Computer and Communications Security*, pages 62–72, 2003.

[5] Technical report Group, T.C.:Trusted Platform Module (TPM) specifications. http://www.trustedcomputinggroup.org/specs/tpm.

[6] Elaine Shi, Adrian Perrig, and Leendert van Doorn. Bind: A fine-grained attestation service for secure distributed systems. In *IEEE Symposium on Security and Privacy*, pages 154–168, 2005.

[7] Stumpf, Omid Tafreschi, Patrick Roder, and Claudia M. Eckert. A robust integrity reporting protocol for remote attestation. In *WATC'06. Second Workshop on Advanced in Trusted Computing*, 2006.

[8] Christoph Krauß, Frederic Stumpf, and Claudia M. Eckert. Detecting node compromise in hybrid wireless sensor networks using attestation techniques. In *ESAS*, pages 203–217, 2007.

[9] Arvind Seshadri, Adrian Perrig, Leendert van Doorn, and Pradeep K. Khosla. Swatt: Software-based attestation for embedded devices. In *IEEE Symposium on Security and Privacy*, pages 272–, 2004.

[10] Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, and Pradeep K. Khosla. Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems. In *SOSP*, pages 1–16, 2005.

[11] Taejoon Park and Kang G. Shin. Soft tamper-proofing via program integrity verification in wireless sensor networks. *IEEE Trans. Mob. Comput.*, 4(3):297–309, 2005.

[12] Diomidis Spinellis. Reflection as a mechanism for software integrity verification. *ACM Trans. Inf. Syst. Secur.*, 3(1):51–62, 2000.

[13] TI MSP-430 processor. http://focus.ti.com/.

[14] Mica2 series. http://www.xbow.com/.

[15] Micheal Mitzenmacher and. Probability and computing: Randomized algorithms and probabilistic analysis. In *Cambridge University Press*, 2005.

[16] Ronald L. Rivest. The rc5 encryption algorithm. In *FSE*, pages 86–96, 1994.

[17] Germano Guimarães, Eduardo Souto, Djamel Fawzi Hadj Sadok, and Judith Kelner. Evaluation of security mechanisms in wireless sensor networks. In *ICW/ICHSN/ICMCS/SENET*, pages 428–433, 2005.

[18] Mica2 wireless measurement system. Crossbow product datasheet, 2003. www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.

[19] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *SenSys*, pages 162–175, 2004.

[20] Bharath Sundararaman, Ugo Buy, and Ajay D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281 – 323, 2005.

[21] Victor Shnayder, Mark Hempstead, Bor-rong Chen, Geoff Werner Allen, and Matt Welsh. Simulating the power consumption of large-scale sensor network applications. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 188–200, New York, NY, USA, 2004. ACM.

[22] Young-Geun Choi, Jeonil Kang, and DaeHun Nyang. Proactive code verification protocol in wireless sensor network. In *ICCSA (2)*, pages 1085–1096, 2007.

978-1-4244-4148-8/09/$25.00 ©2009