MoteSec-Aware: A Practical Secure Mechanism for Wireless Sensor Networks

Yao-Tung Tsou, Chun-Shien Lu, *Member, IEEE,* and Sy-Yen Kuo, *Fellow, IEEE*

***Abstract*—Ensuring the security of communication and access control in Wireless Sensor Networks (WSNs) is of paramount importance. In this paper, we 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 (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, we investigate the Key-Lock Matching (KLM) method to prevent unauthorized access. We implement MoteSec-Aware for the TelosB prototype sensor platform running TinyOS 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.**

***Index Terms*—Sensor networks, security.**

I. Introduction

A. Background

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/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, we 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. In fact, we are aware that a secure mechanism suitable for wireless sensor networks has not been constructed yet, as the related works reviewed below indicate.

B. Related Work

As for the secure network protocol for WSNs, some technologies have been developed, including SPINS [1], TinySec [2], ZigBee [3], and MiniSec [4]. TinySec and ZigBee, which belong to a kind of secure sensor network link-layer protocol, enjoy significant attention in the community. Nevertheless, TinySec achieves low energy consumption by reducing the level of security provided. In contrast, ZigBee satisfies high security, but suffers from high energy consumption. SPINS, on the other hand, achieves low energy consumption by keeping a consistent counter† 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 [10] and FlexiSec [11], all focus on secure network protocol and do not consider the security of data stored in nodes. Kun *et al.*'s method [5] 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. Although R-LEAP+ [12] provided an approach to resist replay and jamming attacks, it is too computation-intensive to be practical for resource-limited sensor network.

In addition to secure network protocol, the issue of secure data storage receives considerable attention at all times. Recently, technologies for secure data storage have been developed not only for social networks (or cloud networks) but also for sensor networks in view of the need of privacy preserving [7] [8] [9]. In contrast to privacy-preserving, we focus on the authority of accessing the stored data in this paper. Although He *et al.* proposed an access control mechanism using ring signature [30], it is not suitable for resource-limited sensor network because the signature scheme may cost higher energy consumption.

C. Overview of Our Method and Contributions

We 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 1.X operating system. In fact, MoteSec-Aware provides (1) a secure network protocol to permit data transmitted in an encrypted format in the air and (2) 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.

†The terms "counter" and "count" are used interchangeably in this paper.

Manuscript received June 26, 2012; revised November 1, 2012 and January 8, 2013; accepted April 6, 2013. The associate editor coordinating the review of this paper and approving it for publication was Y. Chen.

Y.-T. Tsou and S.-Y. Kuo are with the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC (e-mail: yaodong@iis.sinica.edu.tw, sykuo@cc.ee.ntu.edu.tw).

C.-S. Lu is with Institute of Information Science, Academia Sinica, Taipei, Taiwan, ROC (e-mail: lcs@iis.sinica.edu.tw). C.-S. Lu is the contact author.

Digital Object Identifier 10.1109/TWC.2013.050313.120929

1536-1276/13$31.00 © 2013 IEEE

More specifically, we 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 [10] [13]. We present a Virtual Counter Manager (VCM) with synchronized incremental counters and explore the Key-Lock Matching (KLM) method [14] to, respectively, resist the replay/jamming attacks and achieve memory data access control. On the other hand, since sensors in the network, particularly those with limited resources, may suffer from DoS attacks, our previous work, called Constrained Function-based Authentication (CFA) [28], is employed with proper modification to resist DoS attacks. We denote the process of executing CFA in the AES with Offset Codebook Mode (OCB) mode[‖] as AES-OCFA. In this paper, AES-OCFA is the approach proposed to achieve the goal of secure network protocol.

On the other hand, Memory Data Access Control Policy (MDACP) is presented to achieve the goal of data access control. To defend against unauthorized users in accessing data, we investigate the Key-Lock Matching (KLM) method [14] to define access rights in each node because of its characteristic in needing low computation overhead (which will be later described in Section III-C). In KLM, each user is associated with a key ($e.g.$, a prime number) and each file is associated with a lock value. 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. Here, data access control is designed exclusively for function nodes. Note that since the prior methods [1] [2] [3] [4] were not designed for two-tiered networks, which allow leader node or mobile node to query data through an on-demand communication link to some function nodes, and did not allocate the memory zone to utility data, appliances, $etc.$, they cannot directly apply KLM for access control.

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.
- 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.

The rest of the paper is organized as follows. Section II describes the system model, including network topology and attack model. In Section III, we describe the details of the proposed method. In Section IV, evaluation of the maximum counter synchronization error (MCSE) is described. The metrics used to evaluate and analyze the performance for MoteSec-Aware are described in Section V. Experimental and simulation results are described in Section VI. Finally, the conclusions are drawn in Section VII.

[‖] Among authenticated-encryption schemes, it is known that the OCB mode [15] is about two times faster than the CBC-MAC mode. Therefore, OCB is suitable for WSNs.

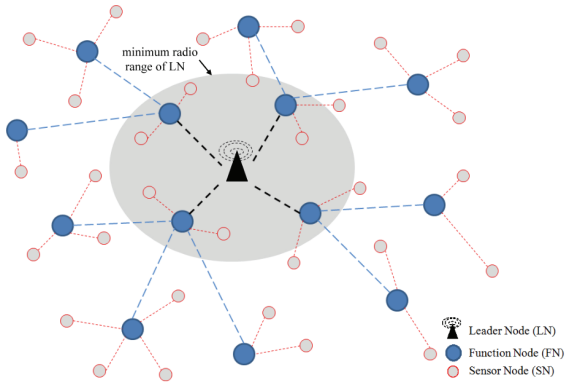


Fig. 1. MoteSec-Aware network topology.

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 (LN), function node (FN), and sensor node (SN), in our sensor network topology. They are classified according to their hardware resources (remaining energy, memory size, *etc*.) [19]. 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. In each cluster, SNs are responsible for collecting sensed data, while FNs aggregate the data from SNs; send commands to SNs; keep utility data, appliances, *etc*. in inside memory; and forward the received data to their upper level nodes (*i.e.*, LNs, FNs). The LN is a network owner with abundant resources that can query data by an on-demand wireless link connected to all FNs. To prevent storage overflow of FNs, the LN can also be periodically dispatched to collect data and empty the storage of 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 (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.

- **Data Confidentiality:** This is the basic property of a secure communication protocol in that data should be kept secret from unauthorized reading.
- **Replay and Jamming Detection:** Communication data should be ensured to be recent and verified that an adversary does not replay or jam data.
- **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.
- **DoS-Resilience:** The DoS attacks, aiming to deplete energies, must be resisted in particular for resource-limited sensor nodes.
- **Data Access:** The adversary should be detected and prevented from accessing data stored on nodes.

C. Key Distribution and Update

Since TinySec, ZigBee, and MiniSec are free to use keying mechanism, they are also free to use any key distribution and update mechanism. SPINS and Kun *et al.*'s method adopt one-way key chain and stateless group key update, respectively, to address the problem of key distribution and update.

Similarly, we discuss the distribution and update of secret keys in MoteSec-Aware in this section. To keep the confidentiality of messages transmitted over the network, there are two types of keys, session keys (used for LN/FNs to broadcast packet to FNs/SNs) and pairwise keys (used for each pair of nodes), used in our system. Here, the session key is distributed in advance. After sensor deployment, pairwise keys are constructed for pairs of sensor nodes by applying our CARPY+ scheme [23]. The advantage of CARPY+ is that it can establish a pairwise key between each pair of sensor nodes without needing any communication. This property is essential in constructing the CFA scheme, because establishing a key via communication incurs an authentication problem, leading to circular dependency. CARPY+ is also resilient to a large number of node compromises so that the complexity for breaking the CARPY+ scheme is $\Omega(2^{\ell+1})$, where ℓ is a security parameter independent of the number of sensor nodes. When updating the session keys, we customize stateless session keys update schemes, which organize one-way key chain to facilitate the authentication of future keys based on previous ones. In stateless session keys update scheme, network owner α uses the pairwise key $K_{\alpha,\beta}$ shared with each non-revoked node β to encrypt the new session key.

D. Comparisons with State-of-the-Art Methods

Our system, MoteSec-Aware, is a fully-implemented general purpose security mechanism for a WSN. The comparisons among some well-known methods and our proposed method, which are common in aiming to manage an IV, are shown in Table I. We can observe that (1) the proposed AES-OCFA strategy enables one to detect replay and jamming attacks based on synchronized incremental counter; (2) while some prior works did not deal with DoS-resilience and memory data access control, the proposed AES-OCFA and MDACP strategies possess these functionalities; and (3) our method incurs no packet overhead and consumes communication cost less than prior works.

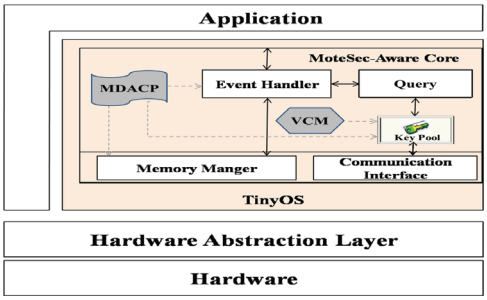


Fig. 2. MoteSec-Aware protocol stack.

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 ($e.g.$, the device drivers). These two layers provide all the basic services and components of the limited available resources ($e.g.$, clock and radio), 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 (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 ($i.e.$, AES) with a communication key ($i.e.$, session key or pairwise key) 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. More specifically, included in AES-OCFA are two procedures for mitigating DoS and detecting replay/jamming attacks. Basically, in order to tackle DoS, our previously proposed method Constrained Function-based Authentication (CFA) scheme [28] has been properly modified and incorporated with the AES in OCB mode. Note that AES in OCB mode generates a ciphertext that simultaneously provides data secrecy and authenticity. In addition, CFA with AES in OCB mode is more efficient than CFA with AES in CBC-MAC mode since the OCB mode is about two times faster than the CBC-MAC mode. Therefore, the modification encourages our system to be more powerful.

A. Security Against DoS via CFA with AES in OCB Mode

The DoS attacks ($i.e.$, false data injection attack and path-based denial of service (PDoS) attack) 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

TABLE I
COMPARISONS OF OUR METHOD WITH SOME STATE-OF-THE-ART METHODS (N: NUMBER OF NODES; φ: PACKET LOSS RATE; I: BYTES OF THE IV)

	Replay Detection	Jamming Detection	DoS Resilience	Memory Access Control	Packet Security Overhead	Communication Cost
SPINS [1]	Yes	Yes	No	No	Counter resynchronization	$O(N\varphi^c)(c: constant)$
TinySec [2]	No	No	No	No	With 8-byte IV	$O(N+NI)$
Zigbee [3]	Yes	No	No	No	With 8-byte IV	$O(N+NI)$
MiniSec [4]	Yes	Yes	No	No	Few bits of the IV	$O(N+NI)$
MoteSec-Aware (our method)	Yes	Yes	Yes	Yes	No	$O(N)$

have been many authentication schemes proposed for wireless sensor networks. However, they are not as efficient in energy consumption as the CFA scheme that we proposed in [28]. In particular, CFA is the first authentication scheme supporting en-route filtering with only a single packet overhead.

In the CFA scheme, the network planner, before sensor deployment, selects a secret polynomial $f(x,y,z,w)$ from the constrained function set ξ [28], whose coefficients should be kept secret, thereby constituting the security basis of CFA. Please refer to [28] for the generation of the constrained function set ξ via the off-line procedure of CFA. For simplicity, we assume that the degree of each variable in $f(x,y,z,w)$ is the same, although they can be distinct in our scheme. For each node u, the network planner constructs two polynomials, $f_{u,1}(y,z,w)=f(u,y,z,w)$ and $f_{u,2}(x,z,w)=f(x,u,z,w)$. Since directly storing these two polynomials enables the adversary to obtain the coefficients of $f(x,y,z,w)$ by capturing a few nodes, the authentication polynomial $auth_u(y,z,w)$ and verification polynomial $verf_u(x,z,w)$ should be constructed from the polynomials $f_{u,1}(y,z,w)$ and $f_{u,2}(x,z,w)$, respectively, by adding independent perturbation polynomials. Afterwards, the authentication and verification polynomials are stored in node u. For source node u, the MAC attached to the message msg is calculated according to its own authentication polynomial. Let the verification number be the result calculated from the verification polynomial $verf_u(x,z,w)$ by applying the claimed source node ID, the shared pairwise key, and the hashed message into x, z, and w, respectively. The receiver considers the received message authentic and intact if and only if the verification difference, which is the difference between the MAC and its calculated verification number, is within a certain predetermined range.

Here, CFA is slightly modified and incorporated with AES in OCB mode within MoteSec-Aware to provide DoS-resilience. The on-line procedure of modified CFA is described in Algorithm 1. In order to incorporate CFA in our system, AES in OCB mode takes msg, $K_{v,u}$, and an IV as inputs, and generates the ciphertext $E_{K_{u,v},IV}(msg)$ and hash value $h(msg)$. It should be noted that the pairwise key $K_{u,v}=K_{v,u}$ is constructed by employing our CARPY+ scheme [23] on nodes u and v, respectively. IV is involved as an input in the encryption operation $E_{K_{u,v},IV}(\cdot)$ such that different ciphertexts can be generated from the same plaintext in order to satisfy semantic security. Note also that AES in OCB mode generates a ciphertext that simultaneously provides data secrecy and authenticity. In the following, we describe the operations one should perform when the node acts in different roles.

Source Node (Message Transmission): When node u wants to send a message msg to the destination node v, it calculates the message authentication code (Step 4):

$$MAC_u(v,msg)=auth_u(v,K_{u,v},h(msg))+n_{u,s},$$

where $n_{u,s}$, used for perturbation, is randomly selected from the set $[0,2^{r-2}]$. Moreover, $h(msg)$ is a value that is generated based on AES in OCB mode. Then, the packet $M=<Header,u,v,E_{K_{u,v},IV}(msg),MAC_u(v,msg)>$ is sent to v possibly through a multihop path (Step 5). Note that the message authentication code $MAC_u(v,msg)$ is only a number [28].

Destination Node (Message Verification): After receiving the packet M, the destination node v first calculates the verification number $verf_v(u,K_{v,u},h(msg))$ according to its own verification polynomial $verf_v(x,z,w)$ and calculates the corresponding *verification difference*, $VD_{v,u}$, as (Step 19):

$$VD_{v,u}=|verf_v(u,K_{v,u},h(msg))-MAC_u(v,msg)|.$$

If $VD_{v,u}\in[0,2^{r-1}-1]$, then the authenticity and integrity of the packet M is successfully verified. Otherwise, the packet M is dropped (Steps 20-26).

The verification process (Steps 7-16) on an intermediate node is the same as the one on the destination node. If the authenticity of the packet M is successfully verified, then the packet M will be forwarded by the intermediate node; otherwise, the packet M is dropped.

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 a 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, δ, between any pair of nodes. In other words, when the packet transmission time is much longer than δ, the jamming attack can be detected at receiver. We will later show in Section IV that δ can be accurately estimated. If a packet does not suffer the jamming attack, the receiver applies

[§]The buffer filter is a data structure used for membership tests: it requires a two-dimensional array of $N^{'}*2$ bits and features two columns, *i.e.*, source address (SrcAddr) and initialization vector (IV), where $N^{'}$ is the number of SNs in a cluster.

Algorithm 1: Modified-CFA-On-Line-Step(M)

Scenario: node u sends a packet M to node v
Input: packet M
Output: result of operation

Source Node u :
Calculate $K_{u,v}$ based on **CARPY+**;
Calculate $E_{K_{u,v},IV}(msg)$ and $h(msg)$ by taking AES in OCB mode;
Compute $MAC_u(v, msg) = auth_u(v, K_{u,v}, h(msg)) + n_{u,s}$, where $n_{u,s}$ is randomly picked from $[0, 2^{r-2} - 1]$;
Send the packet $M = \langle Header, u, v, E_{K_{u,v},IV}(msg), MAC_u(v, msg)\rangle$;
Return $TRUE$;
Intermediate Node ϵ **(on receiving** M**)** :
Calculate $K_{u,\epsilon}$ and $h(msg)$;
Calculate $VD_{\epsilon,u} = |verf_\epsilon(u, K_{\epsilon,u}, h(msg)) - MAC_u(v, msg)|$;
if $VD_{\epsilon,u} \in [0, 2^r - 1]$ **then**
  Forwarding M;
  Return $TRUE$;
else
  Drop M;
  Return $FALSE$;
end
Destination Node v **(on receiving** M**)** :
Calculate $K_{u,v}$ and $h(msg)$;
Calculate $VD_{v,u} = |verf_v(u, K_{v,u}, h(msg)) - MAC_u(v, msg)|$;
if $VD_{v,u} \in [0, 2^{r-1} - 1]$ **then**
  Accept M;
  Return $TRUE$;
else
  Drop M;
  Return $FALSE$;
end

a buffer filter to detect whether the packet suffers the replay attack. We provide evidence for this claim by carrying out a detailed empirical evaluation on TelosB motes.

Now, we describe the synchronized incremental counter approaches at both the sender and receiver sides, as well as the operation of counter synchronization between the sender and receiver in the following.

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.

Receiver Side: The receiver node will receive an incoming packet after propagation delay [21] 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. The proposed synchronized incremental counter approach at

Algorithm 2: Sender side: synchronized incremental counter approach

Scenario: node u sends a message msg to node v
Input: IV (from sender's VCM) and $K_{u,v}$
Output: the packet that processed via AES-OCFA

if *radio channel* = *success* **then**
  Send out the packet $\langle Header, u, v, E_{K_{u,v},IV}(msg), MAC_u(v, msg)\rangle$;
else
  Back off for a random period of time and then go to the step 1;
end

the receiver side is depicted in Algorithm 3. While receiving a packet, the receiver firstly executes Algorithm 1 (Modified-CFA-On-Line-Step) to verify whether it incurs DoS attacks (Step 1). If the verification via Modified-CFA-On-Line-Step algorithm returns "$TRUE$," which means there does not suffer DoS attacks, then the packet is checked to see whether it has been replayed or jammed. Subsequently, the receiver gets a current counter value (CCV) from VCM and calculates a range counter interval (*RCI*)=$[CCV - \delta + 1, CCV]$ (Step 2). RCI is used as a set of IVs to verify the received packet. If all decryptions (via the operation AES.decrypt() in Step 5) fail within the interval defined in RCI, then the packets may be jammed or invalid ($e.g.$, severe fading). Under this circumstance, such packets are dropped (Steps 7-9). In order to detect replay attacks, we simply use a buffer to filter out duplicate packets. The receiver queries the corresponding buffer filter for the tuple *(SrcAddr, IV)* of a packet. If the query returns "$success$," which means there actually does not have the duplicate tuples in the buffer filter, then the packet is considered to be a non-replayed packet and is consequently added into the buffer filter (Steps 11-12). Otherwise, the packet is treated as a replay and is consequently dropped (Steps 13-15).

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 [26] enable pairwise time synchronization with error of mere μs. Transmission delay between neighboring nodes are on the order of ms. Thus, we launch VCM to synchronize counter value based on Secure Pairwise Synchronization (SPS) protocol [26]. Note that the protocol is modified to conform to the security properties addressed in MoteSec-Aware and the resultant pairwise counter synchronization (PCS) protocol is depicted in Algorithm 4.

As shown in Algorithm 4, node A sends a synchronization packet to B at clock $\mathcal{C}_1$ and node B receives this packet at $\mathcal{C}_2$ (Step 1). At clock $\mathcal{C}_3$, then, B sends back an acknowledgement packet (Step 2). This packet contains the values of $\mathcal{C}_2$ and $\mathcal{C}_3$. When node A receives the packet at $\mathcal{C}_4$, it can now calculate the end-to-end counter delay, C_d (Step 3). In PCS, a jamming attack is detected through a comparison (Steps 4-9) of C_d with δ.

In the proposed PCS algorithm (Algorithm 4), message

Algorithm 3: Receiver side: synchronized incremental counter approach

Scenario: node v receives a packet M from node u
Input: RCI (from receiver's VCM)
Output: result of verification

if *Modified-CFA-On-Line-Step*$(M)= TRUE$ **then**
  Compute range counter interval (RCI)= $\{x|(CCV-\delta+1)\leq x\leq CCV\}$;
  Set index= 0;
  repeat
    if *AES.decrypt(RCI[index])*$\neq$ *success* **then**
      index++;
      if *index*= δ **then**
        Drop the jammed packets out;
      end
    else
      if *check buffer-filter(SrcAddr,RCI[index])= success* **then**
        Store (SrcAddr,RCI[index]) into buffer filter;
      else
        Drop the replayed packets out;
      end
      index= δ;
    end
  until $index\neq\delta$;
end

integrity and authenticity are ensured through the use of *MAC*, and of a $K_{A,B}$ (= $K_{B,A}$) that is shared between A and B (Steps 2 and 4). This prevents external attackers from successfully modifying any values in the synchronization process. Furthermore, the adversary cannot impersonate node B as it does not know the secret key $K_{A,B}$. Replay attacks are avoided by using an IV during the handshake.

Algorithm 4: Pairwise Counter Synchronization (PCS)

Input: IV (from random function) and $K_{A,B}$
Output: result of synchronization

A($\mathcal{C}_1$)→($\mathcal{C}_2$)B: synchronization packet $\langle Header, A, B, E_{K_{A,B},IV}(sync), MAC_A(B, sync)\rangle$;
B($\mathcal{C}_3$)→($\mathcal{C}_4$)A: acknowledgement packet $\langle Header, B, A, E_{K_{B,A},IV}(\mathcal{C}_2||\mathcal{C}_3||ack), MAC_B(A, \mathcal{C}_2||\mathcal{C}_3||ack)\rangle$;
Calculate counter delay $C_d = \frac{(\mathcal{C}_2-\mathcal{C}_1)+(\mathcal{C}_4-\mathcal{C}_3)}{2}$;
if $C_d\leq\delta$ **then**
  Counter offset $\Delta = \frac{(\mathcal{C}_2-\mathcal{C}_1)-(\mathcal{C}_4-\mathcal{C}_3)}{2}$;
  Node A updates its counter ($\mathcal{C}_\alpha$): $\mathcal{C}_\alpha = \mathcal{C}_\alpha + \Delta$;
else
  The jamming attack is detected and the received packet is dropped.
end

C. Memory Data Access Control Policy (MDACP)

In order to efficiently secure information in storage and defend against unauthorized users accessing data, we apply KLM to realize MDACP. Here, the unauthorized user is a mobile device/node with radio chipset CC2420 transceiver. In the network, personal information, key materials, and other information that have security concerns will be encrypted by AES-OCFA and stored in the inside memory. In MDACP, each user is associated with a key ($e.g.$, a prime number) and each file is associated with a lock value. 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. MDACP not only stores encrypted files in nodes but also binds the user keys and specific encrypted files together. This approach has greatly reduced the risk of cracking the keys by attacking the encrypted files. In addition, by employing the KLM method, whenever a new user or file is joined, the corresponding key values and lock values will be determined immediately without changing any previously defined keys and locks. This scalability characteristic motivates us to employ KLM for the design of MDACP.

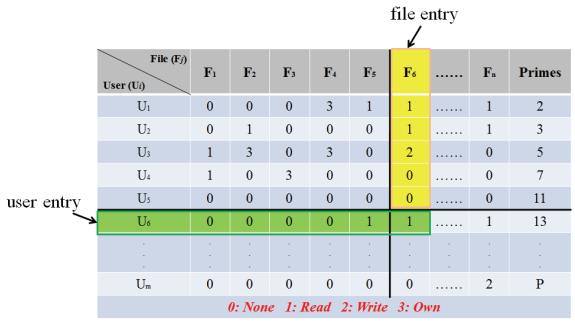


Fig. 3. An example of access right matrix, where the access rights are random numbers. In this case, the access rights are set as follows: 0: None; 1: Read; 2: Write; 3: Own.

1) Algorithm: Let m be the number of users and n be the number of files. We assign a key (K_i, $1\leq i\leq m$) to each user and a lock (L_j, $1\leq j\leq n$) to each file. Let K_i be a prime number and $[r_{ij}]_{m*n}$ be an access right matrix. An example of access right matrix is shown in Fig. 3, where the digit in an entry indicates an access right r_{ij}, which defines the right of user U_i in accessing the file F_j. L_j is then computed by $L_j = \prod_{i=1}^{m} K_i^{r_{ij}}$. To figure out access rights r_{ij}'s of users to files, a function f of key K_i and lock L_j is used. Mathematically, $f(K_i, L_j) = r_{ij}$.

If a user (U_i) asks to access a file (F_j), whether or not the user is legitimate will be verified. The procedure of MDACP is depicted in Algorithm 5, which is used as $f(K_i, L_j)$ to figure out access rights r_{ij}'s from keys and locks. As shown in Algorithm 5, the overhead is dependent on the lock values instead of the number of deployed sensor nodes. Thus, MDACP is efficient for a large-scale sensor network. The experiments are shown in Section VI-A3 for further evidence.

2) Scalability: In sensor networks, energy consumption is a critical issue. As a file/user can be added to or removed from the system without much effort, KLM fits for using on resource-limited sensors. In addition, all operations in MDACP can be implemented simply by recalculating lock values of the corresponding files. These operations will be described as follows.

Algorithm 5: MDACP

Input: Key value K_i of user U_i and lock value L_j of file F_j; $1 \leq i \leq m$ and $1 \leq j \leq n$.

Output: Access rights r_{ij}'s.

1 Set $r_{ij} = 0$ and $Temp$= L_j.
2 Calculate $Q = Temp/K_i$. If Q is an integer, set $r_{ij} = r_{ij} + 1$, $Temp = Q$, repeat this step until Q is not an integer or $r_{ij} = r_{max}$, where r_{max} is the maximum of access right.
3 Output access right r_{ij}.
4 If r_{ij}=Y_{ij}, then execute designate tasks and retrieve corresponding files from the memory; else reject the request.

File insertion/deletion:

- **Insertion of a new file.** When a new file is added to FNs' memories, the LN assigns the access rights of corresponding users to the file and updates the new lock value to all users.
- **Deletion of an old file.** This can be implemented by removing the corresponding file entry from the access right matrices of FNs.

Example: As shown in Fig. 3, if a file F_6 is added to a FN's memory, the LN assigns the "*Read*" right to U_1 and U_2, "*Write*" right to U_3, "*None*" right to U_4 and U_5, and none of the rights to the remainder. Subsequently, the column for F_6 in Fig. 3 is constructed and added to the access right matrix. When the FN receives a query of file access launched by a user, it immediately verifies the user's right by his/her lock value $L_6 = 2^1 * 3^1 * 5^2 * 7^0 * 11^0$. On the contrary, if F_6 is to be removed from the access right matrix, the FN initiates the column related to F_6.

The above processes are not necessary to change the network state. We just add (resp. remove) a file into (resp. from) the FN and update the access right matrix in the FN.

User insertion/deletion:

- **Insertion of a new user.** When a new user, such as mobile device or mobile node, joins in MoteSec-Aware, (s)he has to be verified by the LN in order to obtain a unique user key and lock values. After the new user is successfully verified, the LN updates its information for the FNs' access right matrices.
- **Deletion of an old user.** This can be implemented by directly removing the corresponding user's entry from the access right matrices of FNs.

Example: As shown in Fig. 3, a new user U_6 joins in MoteSec-Aware with user key $K_6 = 13$ and lock values $L_1 = 5^1 * 7^1$, $L_2 = 3^1 * 5^3$, $L_3 = 7^3$, $L_4 = 2^3 * 5^3$, $L_5 = 2^1 * 13^1$, and $L_6 = 2^1 * 3^1 * 5^2 * 13^1$ being assigned by the LN. Then, the LN updates the access rights of U_6 (the row corresponding to U_6 in Fig. 3) for FNs.

We can observe from the above operations that, when a user or file is added to/removed from the network of MoteSec-Aware, LN just sends a packet including the information about a user or file such as user entry or file entry to the FN. Thus, the overhead costs a FN only one packet. Moreover, when the operation of insertion or deletion is conducted for the access right matrix, our proposed protocol takes constant time. This strategy, thus, is efficient in not only communication but also computation.

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

In order to get a norm ($i.e.$, MCSE) for judging whether pulse-delay attacks[††] occur between the sender and receiver, we first carry out an empirical evaluation of the MCSE by measuring the non-malicious end-to-end delay, d, between pairs of nodes. Following the empirical observations found in [24] [25] [26] that d approximately follows a Gaussian distribution, we have $d \propto N(d_{avg}, \sigma^2)$, where d_{avg} denotes average delay and σ denotes standard deviation. Accordingly, d will fall within the interval $[d_{avg} - 3\sigma,\ d_{avg} + 3\sigma]$ with 99.7% confidence, where $d_{avg} - 3\sigma$ denotes the minimum possible delay for the non-malicious setting in both the directions (from sender to receiver and vice-versa).

With the above information available, we then consider how MCSE is affected in the best case for the adversary. Following [26], assuming that the adversary (mote-class adversary) can introduce a maximum pulse-delay factor $\varpi = 12\sigma$. Thus, the sender node can calculate the end-to-end delay in a malicious setting, ε, as:

$$\varepsilon = d_{avg} - 3\sigma + \varpi/2 = d_{avg} + 3\sigma. \quad (1)$$

Now, 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 τ. Since $d_{avg} + 3\sigma$ is the maximum delay, which includes physical layer delay, propagation delay, and others between the sender and receiver, due to d will fall within the interval $[d_{avg} - 3\sigma\ \ d_{avg} + 3\sigma]$ with 99.7% confidence, we have $\tau \leq d_{avg} + 3\sigma$. Under the condition of increasing the synchronized incremental counter by one per d_{avg}, we can calculate the accumulated delay ϑ, involving ε (Eq. (1)) and τ, caused by an adversary as:

$$\begin{aligned} \vartheta &= \lceil (\varepsilon + \tau)/d_{avg} \rceil \\ &= \lceil (d_{avg} + 3\sigma + \tau)/d_{avg} \rceil \\ &\leq \lceil 2 * (d_{avg} + 3\sigma)/d_{avg} \rceil. \end{aligned} \quad (2)$$

As described in Sec. III-B, MCSE is an experiment-based delay counter δ between any pair of nodes. A jamming attack can be detected at receiver if the packet transmission time is much longer than δ. Thus, according to Eq. (2), MCSE is defined to be the upper bound of ϑ as:

$$\delta = \lceil 2 * (d_{avg} + 3\sigma)/d_{avg} \rceil. \quad (3)$$

[††]The adversary jams and replays packets at a later time, thus delaying the receivers' reception [26].

Therefore, the adversary needs to manipulate sufficiently fast and sophisticated hardware to carry out a pulse-delay attack that does not increase the delay by more than δ. If the time required for the packets transmitted in the air exceeds over δ, those packets will be treated to suffer pulse-delay attacks. For theoretical delay analysis, readers can also refer to [32] for details about end-to-end delay upper-bound.

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 σ can be obtained to compute δ according to Eq. (3). 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 [24] [26]. Here, we perform the proposed PCS protocol with the counter being replaced with the timestamp in order to measure d on TelosB motes. The resultant technique is called Pairwise Timer Synchronization (PTS). Our objective is to gauge the distribution of the non-malicious end-to-end delay using timer synchronization, so that an appropriate value δ, 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. Fig. 4(a) shows the actual non-malicious end-to-end delay measured in every run, and Fig. 4(b) shows the distribution of the non-malicious end-to-end delay. The statistics of the non-malicious end-to-end delay over a link shows that the delay falls within the range of [30.9597ms 33.6572ms] with an average delay (d_{avg}) of 32.3244ms and a standard deviation (σ) of 0.6127ms. Subsequently, δ can be calculated according to Eq. (3) as:

$$\delta = \lceil 2 * (d_{avg} + 3\sigma)/d_{avg} \rceil = \lceil 68.3250/32.3244 \rceil = 3. \quad (4)$$

In our experimental environment, the pair-wise distance is different for each pair of nodes. Assuming that the nodes are located in positions wherein they can face at least one other node, the distribution of the non-malicious end-to-end delay is almost equal for all the pairs. This leads to a strong implication, $i.e.$, there is no need to estimate the value of δ within the network during runtime. Instead, it can be calculated before the deployment of the actual network and the nodes can be pre-configured with this value, greatly reducing the overhead. We do note that the non-malicious end-to-end delay will be different for sensor network platforms that use different radios. Thus, δ will vary with different conditions.

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 δ 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 $f(x, y, z, w)$. 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 $\Omega(q^{d+1})$, 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 Pairwise Counter Synchronization

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 $C_A(t) = \lfloor f_A * t \rfloor$ observed at the reference clock at A. Node B with clock frequency f_B can estimate its current frequency error relative to the reference node A's clock as [31]:

$$\frac{(C_A(t_2) - C_B(t_2)) - (C_A(t_1) - C_B(t_1))}{C_A(t_2) - C_A(t_1)}. \quad (5)$$

Let the counts start from zero, *i.e.*, $C_A(t_1) = C_B(t_1) = 0$, and let L be the time duration between t_1 and t_2. For notation simplification, let t_1 start from 0. Eq. (5) can be rewritten as:

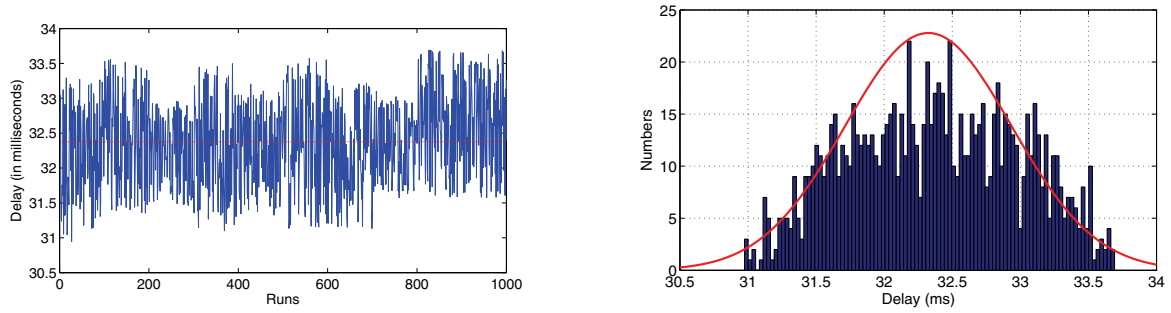
$$\frac{C_A(L) - C_B(L)}{C_A(L)}. \quad (6)$$

Here, we propose the minimum resolution of the frequency error estimation calculated by taking the difference between two measurements separated by one count as $\delta_Q = \frac{C_A(L)-C_B(L)}{C_A(L)} - \frac{C_A(L)-(C_B(L)+1)}{C_A(L)} = \frac{1}{C_A(L)} = \frac{1}{f_A * L}$. Let λ_{min} be the minimum time required to trigger PCS. It must satisfy:

$$\lambda_{min}\delta_Q \geq \Delta T, \quad (7)$$

where the maximum time synchronization error ΔT is defined as $2 * (d_{avg} + 3\sigma)$.

Fig. 5 shows an evaluation of λ_{min} with a broad range of parameters $f_A = 3 \sim 85$kHz and $L = 10 \sim 100$s.



(a) (b)

Fig. 4. Estimation and distribution of non-malicious end-to-end delay. (a) Non-malicious end-to-end delay at each run. (b) Distribution (in bars) of the non-malicious end-to-end delay and the curve of Gaussian with the best-fit parameters ($\mu = 32.3244$ms, $\sigma = 0.6127$ms, confidence=99.7%).

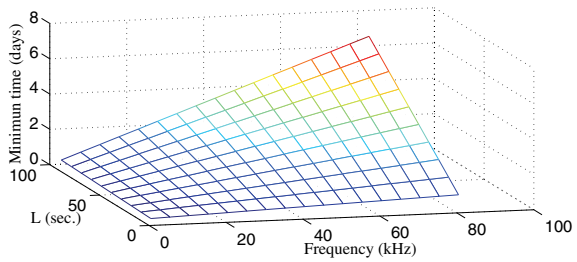


Fig. 5. Minimum time required to trigger the PCS protocol.

We can observe that high-frequency clocks are beneficial to maintain time-synchronization between nodes. For example, for an $f_A = 32$kHz clock frequency in TelosB, if the two beacons are $L = 60$s apart, then the frequency error resolution δ_Q is 0.5ppm (0.000005%). The error over a day can be calculated as $86400 * 0,000005\% = 0.0432$ seconds per day. Therefore, the minimum time required to trigger PCS is $\lambda_{min} \geq 1.6$ days.

D. IV Synchronization for Semantic Security

Achieving semantic security is a critical issue in secure communication, in particular, for the same plaintext that needs to produce different ciphertexts. Recall that IV is involved as an input in the encryption operation $E_{K_{u,v},IV}(\cdot)$. We employ a non-repeating synchronized incremental counter as an IV. The counter is kept synchronized and is increased one per $d_{avg}(\approx 32\text{ms})$. Since we do not append any IV into the packet, the communication overhead is lower than the prior methods, such as MiniSec, TinySec, and SPINS. We set an 8-byte counter as the IV. If the counter maintains the delay epoch that does not exceed δ between the sender and receiver, the IV would be processed over 2^{42} days[¶]. Otherwise, we just launch VCM to synchronize the counter among the nodes.

E. Data Access Control

Following Section III-C1, if an adversary is able to access a file from a node's memory, he/she has the probability of $1/P_R$ in robustly trying valid K_i, L_j, and r_{ij} for an underlying file, where $P_R = m * n * |R|$ and $|R|$ is the size of access right set[‖]. After selecting access rights, LN assigns these values to FNs for constructing access right matrices. Typically, when there is a large-scale network with hundreds of users and files, the successful blind access probability is small enough ($< 10^{-7}$)[‡] to guarantee security. This suffices for the majority of practical applications. On the other hand, the storage overhead for MDACP is $O(mn)$ because it must record an access right matrix $[r_{ij}]_{m*n}$ for verifying each user's right.

F. Energy Consumption

In MoteSec-Aware, the energy consumed for the receiver in receiving a packet from the sender is denoted as E_{total} and is derived as follows. Let δ denote the MCSE at which the node is able to decrypt the packet. Let P_{gar} be the probability that a packet is garbled in transmission and let P_r be the probability that the packet is received. As described in Section III-B, CCV denotes the current counter value that the receiver can read from VCM. Let $\{X(t),\ t \geq 0$ and $CCV - \delta + 1 \leq t \leq CCV\}$ be a counting process if $X(t)$ represents the total number of decryptions that occur by the time t. It can be modeled as a Poisson process with the probability $P(X(t) = k)$ that exactly k decryptions occur for the outcome of n runs being defined as:

$$P(X(t) = k) \quad = \quad (1 - P_{gar})C(n,k)P_r^k(1 - P_r)^{n-k}. \quad (8)$$

Let the Poisson process have rate $\lambda = \frac{nP_r}{t}$. Eq. (8) can be rewritten as:

$$P(X(t) = k) \quad = \quad (1 - P_{gar})\frac{(\lambda t)^k}{k!}e^{-\lambda t}. \quad (9)$$

In fact, δ also indicates the maximum number of trial decryptions and denotes the fault tolerant capability.

Then, the energy consumption per packet is denoted as E_{total} in terms of milliampere second (mAs) and is defined as:

$$\begin{aligned} E_{total} = P(X(t) = k)(X_c E_{AES-OCFA} + E_{VR}) \\ +(1 - P(X(t) = k))(\delta E_{AES-OCFA}), \end{aligned} \quad (10)$$

where X_c denotes the number of decryptions and is set as $0 < X_c \leq \delta$, the cost of a cycle of authentication (for verifying

[¶] $2^{64}(rounds) * 86400^{-1}(day/sec) * 0.032(sec/round) \approx 2^{42}(days)$.

[‖] For example, if LN randomly selects access rights from the access right set $\{0, ..., 2^{10}\}$, then $|R|$ is equal to 1025.

[‡] $1/P_R = 1/(10^2 * 10^2 * (2^{10} + 1)) < 10^{-7}$.

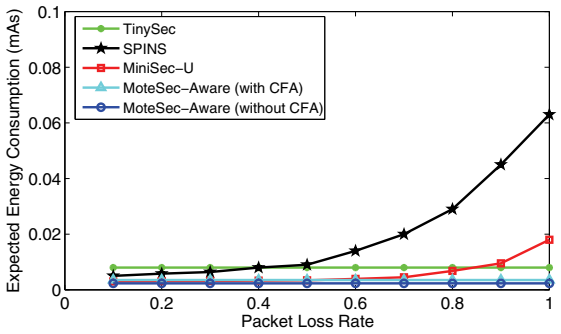


Fig. 6. Packet loss rate vs. energy consumption.

DoS attack and integrity) and decryption by executing AES-OCFA is denoted as $E_{AES-OCFA}$, and E_{VR} is the energy consumed for verifying a replay attack.

Since $X_c \leq \delta$ and $E_{VR} << E_{AES-OCFA}$, E_{total} can be rewritten as:

$$\begin{aligned} E_{total} &\leq P(X(t)=k)(\delta E_{AES-OCFA}+E_{VR}) \\ &\quad +(1-P(X(t)=k))(\delta E_{AES-OCFA}) \\ &\cong P(X(t)=k)(\delta E_{AES-OCFA}) \\ &\quad +(1-P(X(t)=k))(\delta E_{AES-OCFA}) \\ &= \delta E_{AES-OCFA}. \end{aligned} \quad (11)$$

Note that Eq. (11) and our experimental result (shown in Fig. 6) indicates that E_{total} is just a constant even under varying packet loss rates.

Furthermore, the energy consumed for MDACP is spent when events happen. At the reception of an access request, two cases are possible for the receiver: (1) the receiver verifies the access request and permits authorized user to access its memory; (2) an unauthorized user is blocked from accessing the receiver's memory while the request verified by the receiver is not successful. Thus, the energy consumption of data access in MDACP that includes two code sections, the process of KLM and the course of memory access, is defined as:

$$E_{DA} = (1-P_{UA})(E_{KLM}+E_{MA}) + P_{UA}E_{KLM}, \quad (12)$$

where P_{UA} stands for the probability that an unauthorized user wants to access data from a node, E_{KLM} denotes energy consumption of using KLM to verify a request, and E_{MA} is the energy consumption when reading or writing in a node's memory. The derived E_{DA} will be later verified in Section VI-B1 via the TOSSIM simulator [18].

VI. Performance Evaluation

We present a publicly available implementation of MoteSec-Aware for the TelosB platform [16] in Section VI-A and describe TOSSIM-based simulations in Section VI-B.

A. Experiments

The hardware module in MoteSec-Aware is as follows. TelosB is composed of an MSP430 micro controller and a 250kbps 2.4GHz IEEE 802.15.4 TI wireless transceiver. It fully supports TinyOS $1.X/2.X$ and implements a network stack. We implemented our system in nesC code, the programming language used for TinyOS. We have three major code sections: AES-OCFA (AES in OCB mode with CFA scheme), MDACP, and VCM. The ROM needed for our program code is 24756 bytes, and the RAM needed for our program code is 3315 bytes. In the experiments, three issues were examined as follows.

1) Impact of Packet Loss Rate on Energy Consumption: We investigate the impact of the packet loss rate on energy consumption. Fig. 6 shows energy consumption in terms of mAs for TinySec, SPINS, MiniSec-U, and MoteSec-Aware under varying packet loss rates. As the packet loss rate increases, the energy consumption of SPINS presents an exponential rise. As MoteSec-Aware does not require appending an IV to each packet and has a synchronized incremental counter between the sender and receiver, its energy consumption is not affected by the loss of the communication channel. Fig. 6 reveals that MoteSec-Aware with CFA and MoteSec-Aware without CFA cost approximately 0.0023mAs and 0.0035mAs, respectively. In addition, MoteSec-Aware consumes a constant amount of energy and always outperforms other methods.

2) Communication Overhead: We discuss communication overhead in two aspects. First, we analyze the communication overhead of a single packet transmitted between the sender and receiver. Second, we evaluate the traffic overhead for packets transmitted in the network.

For the first problem, the results are shown in Table II. We can see that longer packets are highly costly because of the extra energy consumed by the radio. The results also reveal that MoteSec-Aware and TinyOS have no security overhead increase due to packet communication. In other words, the security overhead of MoteSec-Aware increases over TinyOS is zero; however, those of MiniSec, TinySec, and SPINS are 7.5%, 20%, and 12.5%, respectively. Moreover, both MoteSec-Aware and TinyOS have lower energy consumption than MiniSec, TinySec, and SPINS.

Second, we evaluate the traffic overhead while transmitting packets in the network and observe this packet once per 10 minutes. The traffic overhead generated by our processes ($i.e.$, MoteSec-Aware (without PCS) and MoteSec-Aware (with PCS)) and the traffic overhead related to the accuracy of the counter values in SPINS, MiniSec, and TinySec are, respectively, examined in terms of energy consumption. TinyOS was used as the baseline for comparisons. In MoteSec-Aware (with PCS), we launched PCS once every 10 minutes and estimated the energy consumption, including the cost of transmitting regular data packets and PCS packets. The results are shown in Fig. 7. We have the following observations: (i) The energy consumption of our methods is as low as that of TinyOS but is lower than those of others. (ii) Although SPINS does not need to append an IV to its packets, it consumes more energy in launching immediate counter synchronization. (iii) TinySec consumes most energy because it needs to append an 8-byte IV to its packets. (iv) Although MiniSec only appends a few bits of IV to its packets, it still consumes more energy than our methods because of the additional IV and computation of resynchronization.

3) Overhead of Access Control: The execution time for verifying nodes' rights is shown in Fig. 8. Fig. 8(a) shows the

TABLE II
COMMUNICATION OVERHEAD COMPARISON.

	Payload (Bytes)	Packet Overhead (Bytes)	Security Overhead (Bytes)	Total Size (Bytes)	Energy (mAs)	Increase over TinyOS (%)
TinyOS	28	12	0	40	0.03776	0
SPINS	28	17	5	45	0.04246	12.5
TinySec	28	20	8	48	0.04528	20
MiniSec	28	15	3	43	0.04058	7.5
MoteSec-Aware	28	12	0	40	0.03776	0

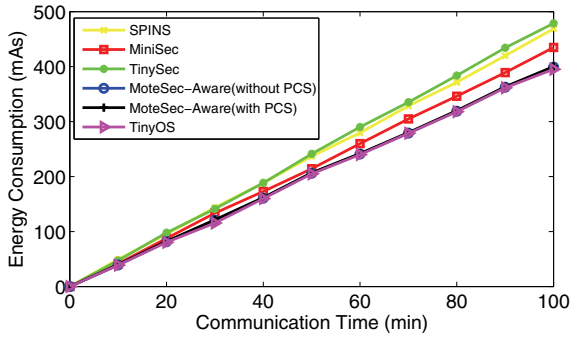


Fig. 7. Communication overhead (communication time vs. energy consumption).

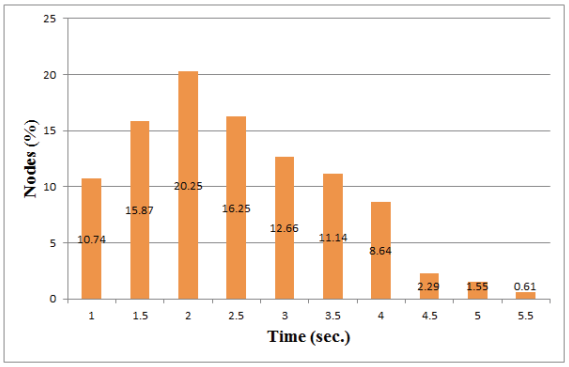


Fig. 9. Completion time of counter synchronization.

time required to verify nodes' rights under variable number of files vs. variable number of sensors. We can observe that the verification time is independent of the number of files and the scale of a network. On the contrary, the verification time is only dependent on lock values and the maximum of access right, as previously described in Algorithm 5. Although the execution time may be prolonged by a large lock value, it is still limited to the maximum of access right. Fig. 8(b) shows the time required to verify nodes' rights under variable number of files vs. variable number of users. We can find that the verification time on TelosB is positively proportional to the number of users but is independent of the number of files.

We estimated the energy consumption of right verification via the formula defined as $E = V * I * t$ [30], where V is the voltage, I is the current, and t is the time duration. TelosB is equipped with two AA batteries, so V is approximately equal to 3 volts. As our result indicates, the execution time depends on the number of users that expect to query the data stored in the nodes' memories; thus, the energy consumption is approximately proportional to the number of users.

B. Simulations

The performance of our method was also simulated in the TinyOS environment with TOSSIM [17] [18] as the WSN simulator. TOSSIM is a discrete-event simulator especially designed for TinyOS operating system to evaluate the energy consumption and the large-scale sensor network operations of MoteSec-Aware.

1) Energy Consumption of AES-OCFA and MDACP: The total simulation time was set to 100 seconds, and 100 nodes were distributed in a sensing field. Each node receives about 10000 packets, and one of the packets is a request to access the data stored in nodes' memories. We obtained the CPU and radio energy consumption associated with each node. The result of energy consumption obtained from our simulation using PowerTOSSIM [18] is that AES-OCFA (3513.5mJ) costs more energy consumption than MDACP (2272.8mJ).

2) Synchronization Time: We tested 100 pairs of nodes to obtain the completion time of counter synchronization. When FNs finish counter synchronization with a trust-based LN, FNs broadcast synchronization packets to their child SNs. For each pair of nodes, we ran 10 rounds of counter synchronization. After nodes finished the counter synchronization, the SNs sent a packet to LN. The trust-based LN calculates the time required to complete counter synchronization. Fig. 9 shows the distribution of completion time of counter synchronization for each individual node. We observe that all nodes have a completion time longer than 1s, but less than 5.5s. In addition, the average completion time for a node is roughly 0.055 second.

VII. CONCLUSIONS

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.

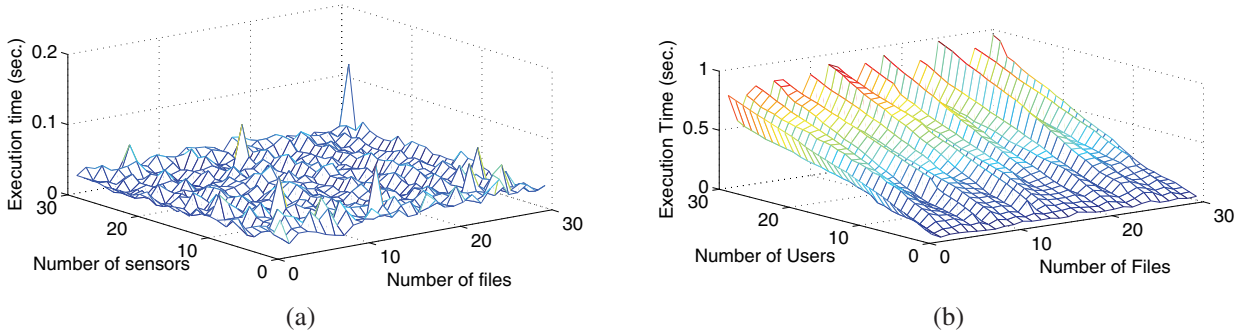


Fig. 8. The execution time for node verification: (a) number of sensors vs. number of files and (b) number of users vs. number of files.

For a large-scale network, strategies such as Bloom Filter [22] may be useful in reducing the storage overhead. This issue will be further studied in the future.

ACKNOWLEDGMENT

Chun-Shien Lu was supported by National Science Council, Taiwan, under Grant NSC 98-2221-E-001-004-MY3. Sy-Yen Kuo was supported by National Science Council, Taiwan, under Grant NSC 99-2221-E-002-108-MY3, and 111 Project under Grant B08002.

REFERENCES

[1] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar, "SPINS: security protocols for sensor networks," in *Proc. 2001 International Conference on Mobile Computing and Networking*, pp. 189–199.

[2] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *Proc. 2004 International Conference on Embedded Networked Sensor Systems*, pp. 162–175.

[3] ZigBee Alliance, Zigbee specifications, Technical Report Document 053474r06, 2005.

[4] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "MiniSec: a secure sensor network communication architecture," in *Proc. 2007 International Conference on Information Processing in Sensor Networks*, pp. 479–488.

[5] S. Kun, L. An, N. Peng, and M. Douglas, "Securing network access in wireless sensor networks," in *Proc. 2009 International Conference on Wireless Network Security*, pp. 261–268.

[6] D. Naor, M. Naor, and J. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Proc. CRYPTO 2001*, pp. 41–62.

[7] X. Lin, X. Sun, X. Wang, C. Zhang, P.-H. Ho, and X. (S.) Shen, "TSVC: timed efficient and secure vehicular communications with privacy preserving," *IEEE Trans. Wireless Commun.*, vol. 7, no. 12, pp. 4987–4998, Dec. 2008.

[8] J. Shi, R. Zhang, and Y. Zhang, "A sspatiotemporal approach for secure range queries in tiered sensor networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 1, pp. 264–273, Jan. 2011.

[9] C. M. Yu, Y. T. Tsou, C. S. Lu, and S. Y. Kuo, "Practical and secure multidimensional query framework in tiered sensor networks," *IEEE Trans. Inf. Forensic and Security*, vol. 6, no. 2, pp. 241–255, 2011.

[10] L. Casado and P. Tsigas, "Contikisec: a secure network layer for wireless sensor networks under the Contiki operating system," in *Proc. 2009 Nordic Conference on Secure IT Systems*, pp. 133–147.

[11] D. Jinwala, D. Patel, and K. Dasgupta, "FlexiSec: a configurable link layer security architecture for wireless sensor networks," *Inf. Assurance and Security*, vol. 4, no. 6, pp. 582–603, 2009.

[12] S. Blackshear and R. Verma, *R-LEAP+: Randomizing LEAP+ Key Distribution to Resist Replay and Jamming Attacks*. ACM Press, 2010, pp. 1985–1992.

[13] NIST, National Institute of Standards and Technology, Computer Security Division, AES standard. Available: http://csrc.nist.gov/archive/aes/index.html, 2001.

[14] J. J. Hwang, B. M. Shao, and P. C. Wang, "A new access control method using prime factorization," *The Computer*, vol. 35, no. 1, pp. 16–20, 1992.

[15] D. Jinwala, D. Patel, and K. S. Dasgupta, "Optimizing the block cipher and modes of operations overhead at the link layer security framework in the wireless sensor networks," in *Proc. 2008 International Conference on Information Systems Security*, pp. 258–272, 2008.

[16] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proc. 2005 International Conference on Information Processing in Sensor Networks*, pp. 364–369.

[17] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proc. 2005 International Conference on Embedded Networked Sensor Systems*, pp. 126–137.

[18] PowerTOSSIM, "PowerTOSSIM: efficient power simulation for TinyOS applications. Available: http://www.eecs.harvard.edu/ shnayder/ptossim/, 2004.

[19] L. Lu, T. He, T. Abdelzaher, and J. Stankovic, "Design and comparison of lightweight group management strategies in EnviroSuite," in *Proc. 2005 International Conference on Distributed Computing in Sensor Networks*, pp. 155–172.

[20] K. Sun, P. Ning, and C. Wang, "TinySeRSync: secure and resilient time synchronization in wireless sensor networks," in *Proc. 2006 International Conference on Computer and Communications Security*, pp. 264–277.

[21] SmartRF, SmartRF CC2420 Datasheet (rev 1.3). Available: http://www.chipcon.com/files/CC2420, Data Sheet 1 3.pdf, 2005.

[22] H. Burton, "Bloom: space/time trade-offs in hash coding with allowable errors," *Commun. of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[23] C. M. Yu, C. S. Lu, and S. Y. Kuo, "Non-interactive pairwise key establishment for sensor networks," *IEEE Trans. Inf. Forensic and Security*, vol. 5, no. 3, pp. 556–569, 2010.

[24] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. 2003 International Conference on Networked Sensor Systems*, pp. 138–149.

[25] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Systems Review - Proc. 2002 Symposium on Operating Systems Design and Implementation*, vol. 36, no. SI, pp. 147–163.

[26] S. Ganeriwal, S. Capkun, and M. B. Srivastava, "Secure time synchronization in sensor networks," *ACM Trans. Inf. and Systems Security*, vol. 11, no. 4, pp. 1–35, 2006.

[27] L. Shu, M. Hauswirth, Y. Zhang, J. Ma, G. Min, and Y. Wang, "Cross layer optimization for data gathering in wireless multimedia sensor networks within expected network lifetime," *J. Universal Computer Science*, vol. 16, no. 10, pp. 1343–1367, 2010.

[28] C. M. Yu, Y. T. Tsou, C. S. Lu, and S. Y. Kuo, "Constrained function-based message authentication for sensor networks," *IEEE Trans. Inf. Forensic and Security*, vol. 6, no. 2, pp. 407–425, 2011.

[29] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *Proc. 1992 International Cryptology Conference*, pp. 471–486.

[30] D. He, J. Bu, S. Zhu, S. Chan, and C. Chen, "Distributed access control with privacy support in wireless sensor networks," *IEEE Trans. Wireless Commun.*, vol. 10, no. 10, pp. 3472–3481, Dec. 2011.

[31] T. Schmid, P. Dutta, and M. B. Srivastava, "High-resolution, low-power time synchronization an Oxymoron No More," in *Proc. 2010 ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 151–161.

[32] Y. Wang, S. Cui, R. Sankar, and S. Morgera, "Delay-throughput trade-off with opportunistic relaying in wireless networks," in *Proc. 2011 IEEE Global Telecommunications Conference*, pp. 1–5.

Yao-Tung Tsou was a research assistant in the Institute of Information Science, Academia Sinica, Taipei, Taiwan, from 2009 to 2012. He is currently working toward the Ph.D. degree at the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan. His research interests include sensor network security and applications.

Chun-Shien Lu received the Ph.D. degree in Electrical Engineering from National Cheng-Kung University, Tainan, Taiwan, Republic of China (ROC), in 1998. After that, Dr. Lu joined the Institute of Information Science, Academia Sinica, Taiwan, as a postdoctoral fellow for his military service. Since March 2013, he has been promoted as a research fellow. His current research interests mainly focus on compressed sensing and sparse representation, multimedia signal processing, and security and privacy-preserving in multimedia and sensor network. Recently, Dr. Lu has served as a Technical Committee member of Multimedia Systems and Applications Technical Committee, IEEE Circuits and Systems Society since 2007. Since 2012, he has served as a Technical Committee member of Communications and Information Systems Security (CIS-TC), IEEE Communications Society. Dr. Lu also serves as Area Chairs of ICASSP 2012, ICASSP 2013, and ICIP 2013. Dr. Lu won Ta-You Wu Memorial Award, National Science Council in 2007 and was a co-recipient of a National Invention and Creation Award in 2004. Dr. Lu is currently an associate editor of IEEE TRANSACTIONS ON IMAGE PROCESSING. He is a member of the IEEE and ACM.

Sy-Yen Kuo is a Distinguished Professor and Dean at the College of Electrical Engineering and Computer Science, National Taiwan University, Taipei, Taiwan and was the Chairman at the Department of Electrical Engineering from 2001 to 2004. He was a Chair Professor and Dean of the College of Electrical and Computer Engineering, National Taiwan University of Science and Technology from 2006 to 2009. He received the BS (1979) in Electrical Engineering from National Taiwan University, the MS (1982) in Electrical & Computer Engineering from the University of California at Santa Barbara, and the PhD (1987) in Computer Science from the University of Illinois at Urbana-Champaign. He spent his sabbatical years as a Visiting Professor at the Department of Computing, Hong Kong Polytechnic University from 2011-2012 and at the Computer Science and Engineering Department, the Chinese University of Hong Kong from 2004-2005, and as a visiting researcher at AT&T Labs-Research, New Jersey from 1999 to 2000, respectively. He was the Chairman of the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan from 1995 to 1998, a faculty member in the Department of Electrical and Computer Engineering at the University of Arizona from 1988 to 1991, and an engineer at Fairchild Semiconductor and Silvar-Lisco, both in California, from 1982 to 1984. In 1989, he also worked as a summer faculty fellow at Jet Propulsion Laboratory of California Institute of Technology. His current research interests include dependable systems and networks, mobile computing, cloud computing, and quantum computing and communications. Professor Kuo is an IEEE Fellow. He has published more than 380 papers in journals and conferences, and also holds more than 15 US and Taiwan patents. He received the distinguished research award between 1997 and 2005 consecutively from the National Science Council in Taiwan and is now a Distinguished Research Fellow there. He was also a recipient of the Best Paper Award in the 1996 International Symposium on Software Reliability Engineering, the Best Paper Award in the simulation and test category at the 1986 IEEE/ACM Design Automation Conference(DAC), the National Science Foundation's Research Initiation Award in 1989, and the IEEE/ACM Design Automation Scholarship in 1990 and 1991.