DOI: 10.2298/CSIS120907025C

Using Bivariate Polynomial to Design a Dynamic Key Management Scheme for Wireless Sensor Networks

Chin-Ling Chen[1], Yu-Ting Tsai[1], Aniello Castiglione[2] and Francesco Palmieri[3]

[1] Department of Computer Science and Information Engineering
Chaoyang University of Technology,
Taichung, 41349, Taiwan
{clc, s10027612}@mail.cyut.edu.tw
[2] Department of Computer Science, University of Salerno
Via Ponte don Melillo, I-84084 Fisciano (SA), Italy
castiglione@ieee.org
[3] Department of Industrial and Information Engineering, Second University of Naples, Via Roma, I-81031 Aversa (CE), Italy
francesco.palmieri@unina.it

Abstract. Wireless sensor networks (WSN) have become increasingly popular in monitoring environments such as: disaster relief operations, seismic data collection, monitoring wildlife and military intelligence. The sensor typically consists of small, inexpensive, battery-powered sensing devices fitted with wireless transmitters, which can be spatially scattered to form an ad hoc hierarchically structured network. Recently, the global positioning system (GPS) facilities were embedded into the sensor node architecture to identify its location within the operating environment. This mechanism may be exploited to extend the WSN's applications. To face with the security requirements and challenges in hierarchical WSNs, we propose a dynamic location-aware key management scheme based on the bivariate polynomial key pre-distribution, where the aggregation cluster nodes can easily find their best routing path to the base station, by containing the energy consumption, storage and computation demands in both the cluster nodes and the sensor nodes. This scheme is robust from the security point of view and able to work efficiently, despite the highly constrained nature of sensor nodes.

Keywords: Sensor Networks, Key Management, Authentication, Bivariate Polynomial Key Distribution.

1. Introduction

WSNs have been deployed in different environments, including disaster relief operations, seismic data collection, monitoring wildlife and battlefield management/military intelligence. Sensors can be installed in a variety of

environments and usually establish a wireless network infrastructure to communicate and exchange information into their operating area. The sensor node is characterized by limited computing power and hence has a low price. Due to their small size, sensors can be spatially scattered to form an ad hoc network. Therefore, WSNs require an appropriate cryptosystem to ensure secure communication and mutual trust between their component nodes. In this scenario, key management becomes an issue of paramount importance since most of the encryption-related primitives require the use and distribution of keys in their operations. The high computational cost of the strongest available techniques (e.g., Diffie-Hellman key management [1] or Rivest Shamir Adleman encryption [2]) make most of them not suitable for use in a WSN, characterized by "hardware-constrained" devices, so that the use of "plain" symmetric cryptography becomes an unavoidable choice. Furthermore, also the key dimension and the number of potentially pre-storable keys may become a significant obstacle to the deployment of strong cryptographic techniques on these tiny devices due to their limited amount of available memory. The last important issue is energy consumption, which is widely known to increase proportionally to the computing efforts [12], such as the ones required by strong cryptosystems. In 2011, Xia et al. [3] focused on addressing the energy efficiency problem in sensor networks.

Accordingly, the use of efficient, lightweight and robust symmetric encryption schemes, together with the associated management protocols, needed to establish and distribute the corresponding keys among the network nodes, assumes a fundamental importance in WSN security. The simplest WSN architectures are based on a strongly meshed, flat interconnection scheme, which is known to exhibit a limited scalability when the number of nodes grows. However, in recent years, starting from the consideration that, in most applications, the connectivity between all the sensors is not necessary, more cost and performance-effective hierarchical schemes [4] are emerging. These schemes, structured according to a multi-tier hierarchical model, allow some "cluster" nodes, characterized by a more powerful hardware equipment (storage and computing capacity, antenna power, battery duration, etc.) to aggregate and pre-process the data incoming from the inexpensive sensor nodes, by reducing the traffic load, the energy consumption as long as the number of hops needed to communicate with the base station (BS), that is in charge for the overall WSN's operations. The most common hierarchical model is two-tier, providing two classes of sensors (basic sensors and cluster aggregators), apart from the BSs.

To cope with this scenario several hierarchical key management and distribution reference schemes have been developed. In detail, Chan, Perrig and Song proposed a Random Key Pre-distribution scheme (RKP) [5], where each node randomly picks *m* keys from a large key pool, such that any two-sensor nodes will share at least one common key with a certain probability. The PIKE scheme [6] addressed the problem of high-density deployment requirements in RKP. Cheng and Agrawal proposed an improved key distribution mechanism known as IKDM [7] using the polynomial keys and easily generating the session keys between the sensor and the cluster nodes.

The cluster nodes use the authentication key to authenticate the sensor nodes. The concept of the large-scale network model underlying IKDM is helpful for us to design our key management scheme. However, the IKDM incurs in high computation cost between the cluster nodes. In this work, we leveraged on the dynamic key management mechanism (KDM) proposed in [8] and on the bivariate polynomial key scheme presented in [7] to establish the session keys and achieve mutual authentication between nodes. The resulting WSN security framework not only can solve the RKP defects (a small number of compromised nodes may expose a large fraction of common keys between the non-compromised nodes [5]), but also successfully copes with the PIKE need to use a lot of sensor memory, and reduces the computation cost of the IKDM. In addition, we also introduced in the resulting key management scheme several location-based considerations and mechanisms [9,10] involving the GPS (Global Positioning system) to identify, in presence of GPS-equipped nodes, the nearest sensors and hence the best key path to the BS on each distribution step.
The rest of this paper is organized as follows. Section 2 reviews some background prerequisites. Section 3 presents our scheme, analyzed in Section 4. Finally, conclusions are presented in Section 5.

2. Backgrounds

2.1. Polynomial Key pre-Distribution Schemes

Polynomial key pre-distribution schemes use the polynomial mathematics in order to generate key pool and perform key assignment among the involved parties. A key distribution server (KDS), performs off-line distribution of several polynomial shares of degree *k* to a set of nodes so that any *k* users are able to calculate a common key that can be used in their communications without any kind of interaction. By evaluating its own stored polynomials with the identifiers (ID) of the other (*k − 1*) parties, each node can determine a common key, independently shared with the other nodes. Blundo et al. [11] proposed a bivariate polynomial $f(x,y)$ that can be used to compute the key; the parameters (x,y) were defined as the unique ID between the sensors x and y respectively. The polynomial is defined as:

$$f(x,y)=\sum_{i,j=0}^{k} a_{ij}x^{i}y^{j} \tag{1}$$

where the coefficients a_{ij} (0 ≤ *i , j* ≤ *k*) are randomly chosen from a finite Galois field *GF(Q)*; *Q* is a prime number that is large enough to accommodate a cryptographic key. The bivariate polynomial has a symmetric property like:

$$f(x,y) = f(y,x) \tag{2}$$

In our specific WSN environment each sensor has a unique ID and, as the first step of network deployment, the KDS first initializes the sensors by giving to each sensor *p* a polynomial share $g_p(y)$, which is obtained by evaluating $f(x,y)$ with $x = p$,

$$g_p(y) = f(p,y) \tag{3}$$

In other words, each sensor node *p* stores a number of *k* coefficients g_j, (0 ≤ *j* ≤ *k*) in its memory,

$$g_j = \sum_{i=0}^{k} a_{ij}(p)^i, (0 \le j \le k) \tag{4}$$

where *p* is the node ID of the sensor, and g_j is the coefficient of y^j in the polynomial $f(p,y)$.

2.2. Properties of Polynomial Key pre-Distribution Scheme

The main strength of the bivariate polynomial key pre-distribution scheme is that there is no overhead during the node-to-node pairwise key establishment activity. The main known drawback, on the other hand, is the "*K*-security" property: a *k*-degree scheme is only robust against coalitions of up to *k* compromised nodes [7]. Until the number of compromised nodes is kept lower than *k*, even if all the compromised nodes share their secret data, the unknown coefficients of the polynomial cannot be calculated. However, when more than *k* nodes are compromised, the coefficients can be determined from the combination of all the available data.

3. The Proposed Scheme

We combine the effectiveness of bivariate polynomial key management schemes with the location based ones. By assuming that the sensor nodes' position can be dynamically determined via GPS, our scheme is also able to leverage on the sensor location information to improve its overall performance, with respect to traditional location-oblivious schemes. It allows data aggregation in a fewer number of places, located on better paths to the BS, by simultaneously achieving the same connectivity and security degree, with a lower number of keys to be stored in each sensor node. The operating phases of the integrated framework are described in the following.

3.1. Notation

The following notation is used in the following.

BS: the base station

CN_i, SN_i: the *i*-th cluster node and the *i*-th sensor node, respectively

$ID_{SN_i}, ID_{CN_i}, ID_{BS}$: the identity of the *i*-th sensor, cluster and BS

$h(.)$: a one-way hash function

$h_{key}(.)$: a one-way hash function with key

$f(x,y)$: a bivariate polynomial, where (x,y) are defined as the unique ID between the sensors x and y

$E_k(M)$: the symmetric encryption making use of key *k* to encrypt *M*

$D_k(M)$: the symmetric decryption making use of key *k* to decrypt *M*

$X \stackrel{?}{=} Y$: determines if *X* equal to *Y*

$Seed$: seed for updating the finish message key pre-deployed in nodes

$a_{i\ i}^{h}, a_i^{h-1}$: two parameters pre-deployed in the *i*-th sensor node for the generation session key (where *h* is an integer of the hash operation)

$b_{i\ i}^{h}, b_i^{h-1}$: two parameters pre-deployed in the *i*-th cluster node for the generation session key (where *h* is an integer of the hash operation)

$SNID_{CN_i}$: the identity list of the sensors served by the cluster node CN_i

$SNKEY_{list}$: the key list of sensor dynamic keys stored to the BS

$SNKEY_{CN_i}$: the key list of the sensors' dynamic keys, generated by the *BS*

K_{SN_i}: the dynamic key of the SN_i, $K_{SN_i} = h(a_i^h, a_i^{h-1})$

K_{CN_i}: the dynamic key of the CN_i, $K_{CN_i} = h(b_i^h, b_i^{h-1})$

$K_{CN_i-CN_j}$: the polynomial session key of the CN_i and CN_j

K_{CN_i-BS}: the polynomial session key of the CN_i and BS

N_{CN_i}: the nonce generated by the BS for the CN_i

MAC_{CN_i-BS}: the message authentication code (*MAC*) for the BS to CN_i

MAC_{BS-CN_i}: the *MAC* for the CN_i to authenticate BS

msg_{CN_i}: the receiving message of the *i*-th cluster node from the decrypted messages of the *p* sensor nodes

$msg_{start}, msg_{finish}$: the start message and finish message, respectively

$msg_{location}$: the location message broadcasted by the cluster node

3.2. Initialization Phase

In this phase, the BS pre-distributes the polynomial scheme parameters to both the sensor nodes and cluster nodes.

Step 1: The BS selects a random number a_i and computes the hash chain:

$$\begin{aligned} a_i^0 &= a_i \\ a_i^1 &= h(a_i^0) \\ a_i^2 &= h(a_i^1, a_i^0) \\ &\vdots \\ a_i^h &= h(a_i^{h-1}, a_i^{h-2}), \quad (1 \le i \le m) \end{aligned} \tag{5}$$

It stores $((a_1^1, a_1^0), ..., (a_m^1, a_m^0))$ to the nodes' dynamic key list $SNKEY_{list}$

$$SNKEY_{list} = ((a_1^1, a_1^0), (a_2^1, a_2^0), ..., (a_m^1, a_m^0)), \quad (1 \le i \le m) \tag{6}$$

It then stores (ID_{SN_i}, (a_i^1, a_i^0), $Seed$, K_{msg}) to the *i*-th sensor node.

Step 2: The BS selects a random number b_i and builds the hash chain as:

$$\begin{aligned} b_i^0 &= b_i \\ b_i^1 &= h(b_i^0) \\ b_i^2 &= h(b_i^1, b_i^0) \\ &\vdots \\ b_i^h &= h(b_i^{h-1}, b_i^{h-2}), \quad (1 \le i \le n) \end{aligned} \tag{7}$$

Then the BS stores $((b_1^1, b_1^0), (b_2^1, b_2^0), ..., (b_n^1, b_n^0))$ in the dynamic key list $CNKEY_{list}$ of the *n* cluster nodes,

$$CNKEY_{list} = ((b_1^1, b_1^0), (b_2^1, b_2^0), ..., (b_n^1, b_n^0)), \quad (1 \le i \le n) \tag{8}$$

The BS randomly selects two polynomials from the *k* ones for *n* cluster nodes, and then stores the bivariate polynomial on these nodes:

$$K_{CN_i - CN_j} = f_{CN_i - CN_j}(ID_{CN_i}, y) \tag{9}$$

$$K_{BS-CN_i} = f_{BS-CN_i}(ID_{BS}, ID_{CN_i}) \tag{10}$$

The BS selects a nonce N_{CN_i}, and then stores $(ID_{CN_i}, (b_i^1, b_i^0), N_{CN_i}, K_{CN_i-CN_j}, K_{BS-CN_i})$ to the *i*-th cluster nodes.

3.3. Location-based Routing Plan Determination

In this phase, the cluster nodes can establish the best route on the basis of the received broadcast location message in a monitoring area.

Step 1: After the Initialization phase, the sensors and cluster nodes have stored the operating parameters and then distributed the associated messages within their operating environment.

Step 2: The BS broadcasts each sensor network start message msg_{start} to the cluster nodes.

Step 3: Upon receiving the start message, the cluster node (equipped with a GPS receiver) broadcasts the message $msg_{location}$ concerning its location to the neighbor cluster nodes.

Step 4: After receiving the message $msg_{location}$, the cluster nodes know the location of the source neighbor cluster so that it can transmit the monitor data to the cluster node that is nearest to the BS.

For example, in Figure 1, the cluster node R_5 can receive the nearest distance message to the BS from the neighbor cluster nodes R_1, R_2, R_3, R_4, R_6, R_7, R_8 and R_9; It can compare the received location message to select the nearest node from the BS and establish the multi-hop routing path to the cluster node R_1. The cluster node R_1 will be used to relay communications to the BS, so the best path of the cluster node R_5 will be established as follows: $R_5 \rightarrow R_1 \rightarrow BS$.

On the basis of the shortest distance between the cluster node and the BS, each cluster node will establish the best routing path. In Figure 2, the cluster node R_9 can determine that the neighbor cluster node on the best path is R_5, and the cluster node R_5 and R_1 can determine the R_1 and BS, respectively. The best path for the cluster node R_9 can be established as follows: $R_9 \rightarrow R_5 \rightarrow R_1 \rightarrow BS$. In the same way, the cluster node R_3 can determine the best path: $R_3 \rightarrow R_2 \rightarrow R_1 \rightarrow BS$.

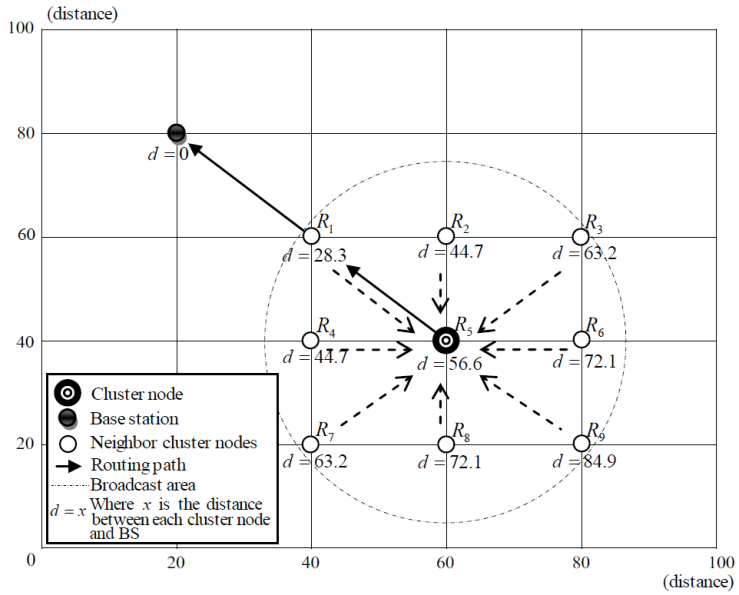

Fig. 1. Each cluster node broadcasts its location to its neighbor cluster nodes

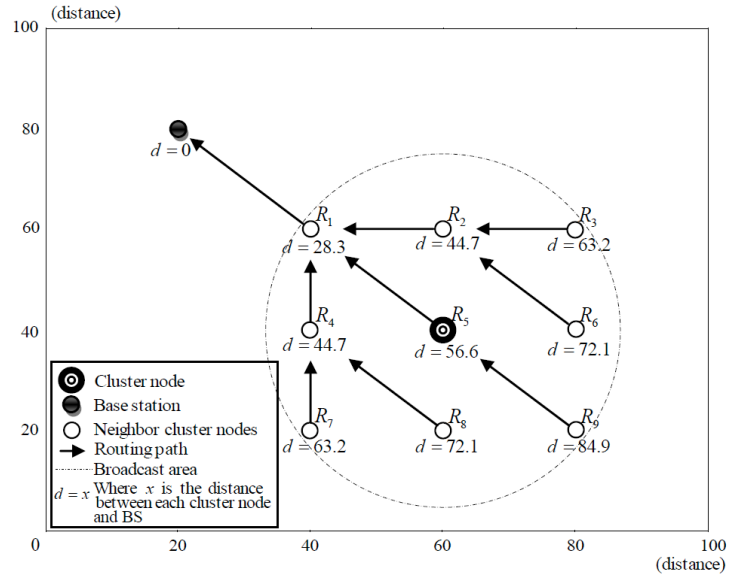

Fig. 2. Establishing the best routing overview

Every pair of nodes along the resulting multi-hop path can establish a pairwise key for encrypted communication in such a way that each intermediate node can relay data towards the BS in a totally secure way. Location awareness

also increases the probability that the geographically closest node pairs establish a pairwise session key along the best path to the BS, with the effect of saving energy on all the nodes involved in multi-hop routing.

3.4. Establishing the Polynomial Session Key Phase

The cluster nodes can use the bivariate polynomial to establish the pairwise session keys along the previously determined multi-hop paths. Each cluster node CN_i broadcasts its unique ID_{CN_i} to the cluster node CN_j and the cluster node CN_j replies with its unique ID_{CN_j} to cluster node CN_i. The cluster nodes receive the related unique ID from the neighbor cluster nodes and compute the session key as follows:

$$K_{CN_i-CN_j} = f_{CN}(ID_{CN_i}, ID_{CN_j}) \tag{11}$$

$$K_{CN_j-CN_i} = f_{CN}(ID_{CN_j}, ID_{CN_i}) \tag{12}$$

3.5. Cluster Node Requests Session Key Phase

Each cluster node, when collecting the monitoring data, receives the associated messages from its *p* sensor nodes, and then decrypts them properly. In order to do this, the cluster node needs to request the session key to the BS. The session key request scenario is shown in Figure 3.

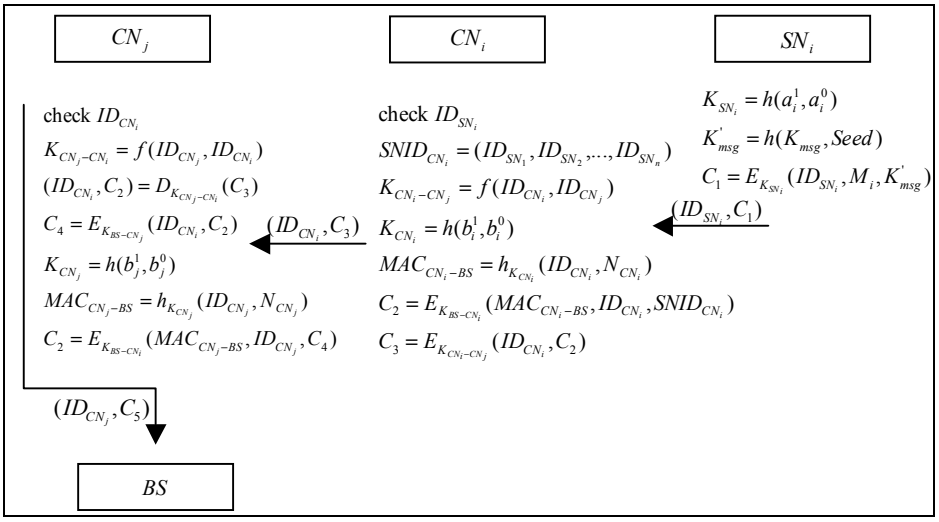

Fig. 3. Overview of the cluster node session key requests

Step 1: First, the sensor node SN_i uses the parameters (a_i^1, a_i^0) to compute the dynamic session key K_{SN_i},

$$K_{SN_i} = h(a_i^1, a_i^0) \tag{13}$$

The sensor node computes the finish message key $K_{msg}^{'}$,

$$K_{msg}^{'} = h(K_{msg}, Seed) \tag{14}$$

Then the sensor node uses the dynamic session key K_{SN_i} to encrypt $(ID_{SN_i}, M_i, K_{msg}^{'})$,

$$C_1 = E_{K_{SN_i}}(ID_{SN_i}, M_i, K_{msg}^{'}) \tag{15}$$

The sensor node then sends (ID_{SN_i}, C_1) to the cluster node.

Step 2: The cluster node CN_i checks ID_{SN_i} and collects the monitoring data of *p* sensor nodes to make an identity list $SNID_{CN_i}$,

$$SNID_{CN_i} = (ID_{SN_1}, ID_{SN_2}, ..., ID_{SN_p}) \tag{16}$$

Then the cluster node CN_i computes the session key $K_{CN_i-CN_j}$,

$$K_{CN_i-CN_j} = f(ID_{CN_i}, ID_{CN_j}) \tag{17}$$

The cluster node CN_i then uses the parameters b_i^1 and b_i^0 to compute the dynamic session key K_{CN_i},

$$K_{CN_i} = h(b_i^1, b_i^0) \tag{18}$$

The cluster node CN_i uses the ID_{CN_i} and nonce N_{CN_i} to compute the message authentication code MAC_{CN_i-BS},

$$MAC_{CN_i-BS} = h_{K_{CN_i}}(ID_{CN_i}, N_{CN_i}) \tag{19}$$

The cluster node CN_i uses the session key K_{BS-CN_i} to encrypt $(MAC_{CN_i-BS}, ID_{CN_i}, SNID_{CN_i})$,

$$C_2 = E_{K_{BS-CN_i}}(MAC_{CN_i-BS}, ID_{CN_i}, SNID_{CN_i}) \quad (20)$$

Then the cluster node CN_i uses the session key $K_{CN_i-CN_j}$ to encrypt (ID_{CN_i}, C_2),

$$C_3 = E_{K_{CN_i-CN_j}}(ID_{CN_i}, C_2) \quad (21)$$

And the cluster node CN_i sends the message (ID_{CN_i}, C_3) to CN_j.

Step 3: Upon receiving the message (ID_{CN_i}, C_3), the cluster node CN_j checks ID_{CN_i} and computes the session key $K_{CN_j-CN_i}$ to decrypt the message C_3,

$$K_{CN_j-CN_i} = f(ID_{CN_j}, ID_{CN_i}) \quad (22)$$

$$(ID_{CN_i}, C_2) = D_{K_{CN_j-CN_i}}(C_3) \quad (23)$$

Then the cluster node CN_j uses the session key K_{BS-CN_j} to encrypt the forwarding message (ID_{CN_i}, C_2) of the cluster node CN_i,

$$C_4 = E_{K_{BS-CN_j}}(ID_{CN_j}, C_2) \quad (24)$$

The cluster node CN_j computes the message authentication code MAC_{CN_j-BS},

$$MAC_{CN_j-BS} = h_{K_{CN_j}}(ID_{CN_j}, N_{CN_j}) \quad (25)$$

And then, the cluster node CN_j encrypts the message $(MAC_{CN_j-BS}, ID_{CN_j}, C_4)$,

$$C_5 = E_{K_{BS-CN_j}}(MAC_{CN_j-BS}, ID_{CN_j}, C_4) \quad (26)$$

and sends the message (ID_{CN_j}, C_5) to the BS.

3.6. Authentication Phase

In this phase, the BS authenticates the cluster nodes. Moreover, the cluster node can also authenticate the BS accordingly. The overview of the authentication phase is shown in Figure 4.

Step 1: Once receiving the message (ID_{CN_j}, C_5), the BS checks the ID_{CN_j} and decrypts C_5,

$$(MAC_{CN_j-BS}, ID_{CN_j}, C_4) = D_{K_{BS-CN_j}}(C_5) \tag{27}$$

Then it computes the message authentication code MAC'_{CN_j-BS} and checks whether or not it is if equal to MAC_{CN_j-BS} ,

$$MAC'_{CN_j-BS} = h_{K_{CN_j}}(ID_{CN_j}, N_{CN_j}) \tag{28}$$

$$MAC'_{CN_j-BS} \stackrel{?}{=} MAC_{CN_j-BS} \tag{29}$$

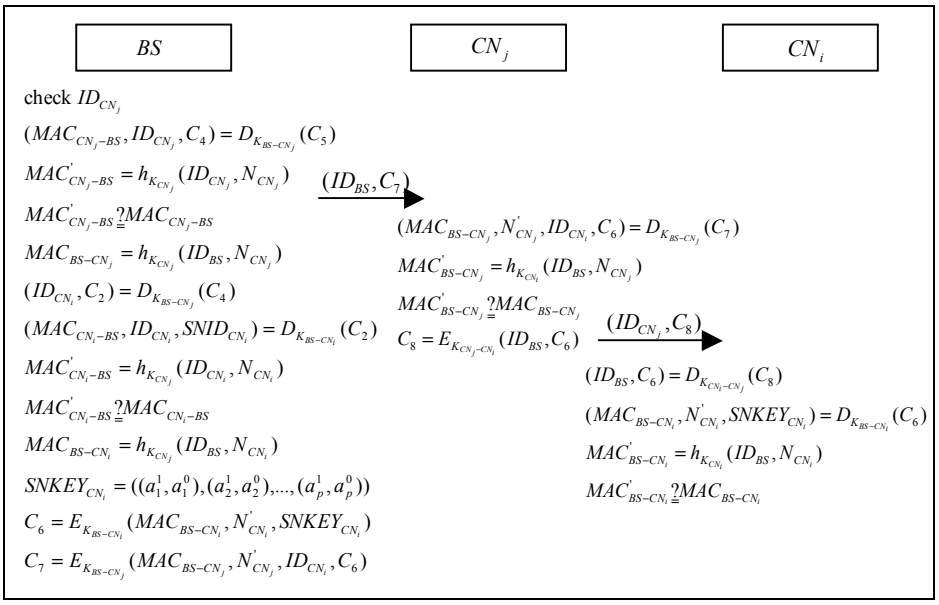

Fig. 4. Overview of the authentication phase

The BS uses the ID_{BS} and nonce N_{CN_j} to compute the message authentication code MAC_{BS-CN_j},

$$MAC_{BS-CN_j} = h_{K_{CN_j}}(ID_{BS}, N_{CN_j}) \tag{30}$$

After that, the BS decrypts the message C_4,

$$(ID_{CN_i}, C_2) = D_{K_{BS-CN_j}}(C_4) \tag{31}$$

Then it decrypts C_2, computes the message authentication code MAC'_{CN_i-BS} and checks whether or not it is equal to MAC_{CN_i-BS},

$$(MAC_{CN_i-BS}, ID_{CN_i}, SNID_{CN_i}) = D_{K_{BS-CN_i}}(C_2) \tag{32}$$

$$MAC'_{CN_i-BS} = h_{K_{CN_i}}(ID_{CN_i}, N_{CN_i}) \tag{33}$$

$$MAC'_{CN_i-BS} \stackrel{?}{=} MAC_{CN_i-BS} \tag{34}$$

Then the BS uses the ID_{BS} and the nonce N_{CN_i} to compute the message authentication code MAC_{BS-CN_i},

$$MAC_{BS-CN_i} = h_{K_{CN_i}}(ID_{BS}, N_{CN_i}) \tag{35}$$

After authentication, the BS uses the ID list $SNID_{CN_i}$ to find the dynamic key of the sensor node in the $SNKEY_{list}$; it then stores the dynamic session key to the key list $SNKEY_{CN_i}$ of the *p* members of the cluster node CN_i,

$$SNKEY_{CN_i} = ((a_1^1, a_1^0), (a_2^1, a_2^0), ..., (a_p^1, a_p^0)), \quad 1 \le i \le p \tag{36}$$

Then the station uses K_{BS-CN_i} to encrypt $(MAC_{BS-CN_i}, N'_{CN_i}, SNKEY_{CN_i})$,

$$C_6 = E_{K_{BS-CN_i}}(MAC_{BS-CN_i}, N'_{CN_i}, SNKEY_{CN_i}) \tag{37}$$

It uses the session key K_{BS-CN_j} to encrypt $(MAC_{BS-CN_j}, N'_{CN_j}, ID_{CN_i}, C_6)$,

$$C_7 = E_{K_{BS-CN_j}}(MAC_{BS-CN_j}, N'_{CN_j}, ID_{CN_i}, C_6) \tag{38}$$

The BS sends (ID_{BS}, C_7) to the cluster node CN_j.

Step 2: Upon receiving (ID_{BS}, C_7), the cluster node CN_j uses the session

key K_{BS-CN_j} to decrypt C_7,

$$(MAC_{BS-CN_j}, N'_{CN_j}, ID_{CN_i}, C_6) = D_{K_{BS-CN_j}}(C_7) \tag{39}$$

Then it computes the message authentication code MAC'_{BS-CN_j} and checks whether or not it is equal to MAC_{BS-CN_j},

$$MAC'_{BS-CN_j} = h_{K_{CN_j}}(ID_{BS}, N_{CN_j}) \tag{40}$$

$$MAC'_{BS-CN_j} \stackrel{?}{=} MAC_{BS-CN_j} \tag{41}$$

The cluster node CN_j uses the session key $K_{CN_j-CN_i}$ to encrypt the message (ID_{BS}, C_6),

$$C_8 = E_{K_{CN_j-CN_i}}(ID_{BS}, C_6) \tag{42}$$

Since the cluster node CN_j has the message ID_{CN_i}, the cluster node CN_j sends the message (ID_{CN_j}, C_8) to the cluster node CN_i.

Step 3: After receiving the message, the cluster node CN_i uses the session key $K_{CN_i-CN_j}$ to decrypt the message C_8,

$$(ID_{BS}, C_6) = D_{K_{CN_i-CN_j}}(C_8) \tag{43}$$

Then the cluster node CN_i decrypts the message C_6 by using the session key K_{BS-CN_i},

$$(MAC_{BS-CN_i}, N'_{CN_i}, SNKEY_{CN_i}) = D_{K_{BS-CN_i}}(C_6) \tag{44}$$

Then it computes the message authentication code MAC'_{BS-CN_i} and checks whether or not it is equal to MAC_{BS-CN_i},

$$MAC'_{BS-CN_i} = h_{K_{CN_i}}(ID_{BS}, N_{CN_i}) \tag{45}$$

$$MAC'_{BS-CN_i} \stackrel{?}{=} MAC_{BS-CN_i} \tag{46}$$

3.7. Dynamic Key Management Phase

The cluster node decrypts the message from the members, collects the monitor data, and sends it to the BS. When the finish message is sent out, the BS and all the sensors update the dynamic key. The overview of the dynamic key management phase is shown in Figure 5.

Step 1: After the authentication, the cluster node CN_i uses the key list $SNKEY_{CN_i}$ to find the dynamic key parameter of *p* members

$$SNKEY_{CN_i} = ((a_1^1, a_1^0),(a_2^1, a_2^0),...,(a_p^1, a_p^0)) \tag{47}$$

The cluster node CN_i gets the dynamic key K_{SN_i} of the sensor nodes; it can decrypt the message C_1 of the monitoring data,

$$K_{SN_i} = h(a_i^1, a_i^0) \tag{48}$$

$$(ID_{SN_i}, M_i, K'_{msg}) = D_{K_{SN_i}}(C_1) \tag{49}$$

After that, the cluster node CN_i gets the message $(M_0, M_1,..., M_P)$ from the decrypted messages of the sensor nodes; it then stores into msg_{CN_i},

$$msg_{CN_i} = (M_0, M_1,..., M_P) \tag{50}$$

Then the cluster node CN_i encrypts $(ID_{CN_i}, msg_{CN_i}, msg_{finish})$,

$$C_9 = E_{K_{BS-CN_i}}(ID_{CN_i}, msg_{CN_i}, msg_{finish}) \tag{51}$$

The cluster node CN_i uses the finish message key K'_{msg} to encrypt the finish message msg_{finish},

$$C_5 = E_{K'_{msg}}(msg_{finish}) \tag{52}$$

The cluster node then sends (ID_{CN_i}, C_9) and (ID_{CN_i}, C_{10}) to the BS and sensor nodes, respectively. The cluster node updates the N_{CN_i} and the dynamic session key K_{CN_i},

$$N_{CN_i} = N'_{CN_i} \tag{53}$$

$$K'_{CN_i} = h(K_{CN_i}, b_i^1), \quad (\text{where } K_{CN_i} = b_i^2) \tag{54}$$

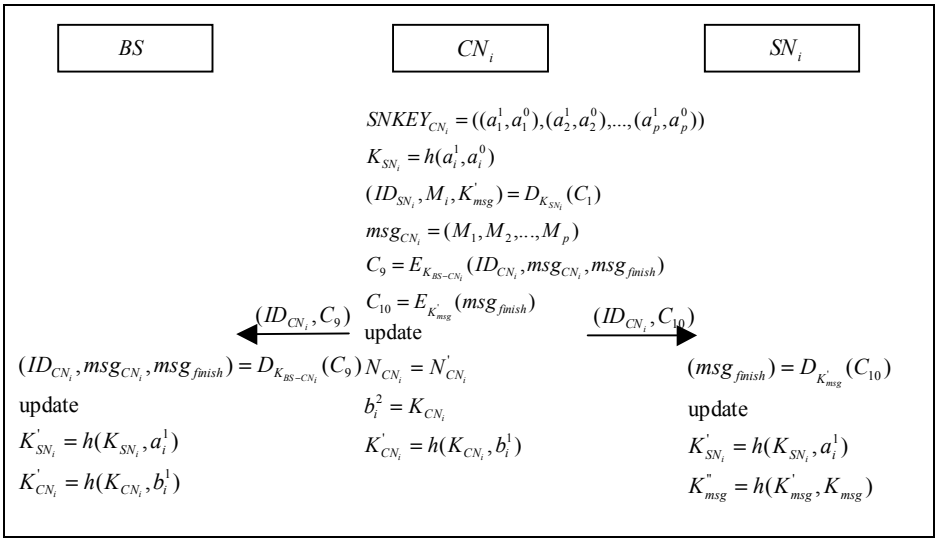

Fig. 5. Overview of the dynamic key management phase

Step 2: Once receiving the message, the BS checks the ID_{CN_i} and decrypts the message C_9 ,

$$(ID_{CN_i}, msg_{CN_i}, msg_{finish}) = D_{K_{BS-CN_i}}(C_9) \tag{55}$$

The BS gets the messages msg_{SN_i} of the sensor nodes, and the finish message msg_{finish}; then the BS updates the dynamic keys K'_{SN_i} and K'_{CN_i} of the sensor node and the cluster node, respectively.

$$K'_{SN_i} = h(K_{SN_i}, a_i^1), \quad (\text{where } K_{SN_i} = a_i^2) \tag{56}$$

$$K'_{CN_i} = h(K_{CN_i}, b_i^1), \quad (\text{where } K_{CN_i} = b_i^2) \tag{57}$$

Step 3: The sensor node SN_i decrypts C_{10} and gets the message msg_{finish} ,

$$msg_{finish} = D_{K'_{msg}}(C_{10}) \tag{58}$$

it updates the dynamic session key K'_{SN_i} and the finish message key K''_{msg}

$$K'_{SN_i} = h(K_{SN_i}, a_i^1), \quad (K_{SN_i} = a_i^2) \tag{59}$$

$$K''_{msg} = h(K'_{msg}, K_{msg}) \tag{60}$$

4. Analysis and Discussion

4.1. Sensor Network Resilience

Since the cluster nodes have the GPS location capability, when the BS sends the start message to the cluster node, the cluster node can find its location, broadcasts the location message to the neighbor cluster nodes and finds the nearest cluster node of the BS. Therefore, if the cluster nodes are compromised or the sensors are at low-energy, the cluster node broadcasts to the new neighbor cluster nodes that are nearest to the BS. In this way, our scheme achieves the network resilience.

4.2. Resistance to Sensor Node Capture Attack

Node capture attack is a serious threat in WSNs deployed in hostile environments. Due to their hardware limitations the nodes usually are not tamper-resistant and hence any adversary that captures a sensor can easily extract its stored secret data to break the underlying security scheme. In our scheme, because the coefficients a_{ij} ($0 \le i, j \le k$) are randomly chosen from a finite *GF(Q)*, where *Q* is a prime number that is large enough to accommodate a cryptographic key, each cluster node pair has a unique pairwise session key $K_{CN_i-CN_j}$ and $K_{CN_j-CN_i}$, so the security cannot be compromised between cluster nodes. if one of them is compromised. Only a legal ID pair ID_{CN_i} and ID_{CN_j} can compute the right session keys:

$$K_{CN_i-CN_j} = f_{CN}(ID_{CN_i}, ID_{CN_j}) \tag{61}$$

$$K_{CN_j-CN_i} = f_{CN}(ID_{CN_j}, ID_{CN_i}) \tag{62}$$

If we use a *k*-degree bivariate polynomial our scheme is guaranteed to be (k + 1)-secure. That is, no less than (k + 1) nodes holding polynomial shares have to be captured in order to reconstruct it.

4.3. Mutual Authentication

We can consider two fundamental cases:

(1) The BS authenticates the *i*-th cluster node

The cluster node uses the dynamic session key to compute the message authentication code MAC_{CN_i-BS} of the N_{CN_i}; it then sends it to the BS.

$$MAC_{CN_i-BS} = h_{K_{CN_i}}(ID_{CN_i}, N_{CN_i}) \tag{63}$$

Since the BS received the message authentication code MAC_{CN_i-BS} from the cluster node CN_i, it can compute the message authentication code MAC'_{CN_i-BS} and checks whether or not it is equal to MAC_{CN_i-BS},

$$MAC'_{CN_i-BS} = h_{K_{CN_i}}(ID_{CN_i}, N_{CN_i}) \tag{64}$$

$$MAC'_{CN_i-BS} \stackrel{?}{=} MAC_{CN_i-BS} \tag{65}$$

(2) The *i*-th cluster node authenticates the BS

The BS uses the dynamic session key to compute the message authentication code MAC_{BS-CN_i} of N_{CN_i}, and sends it to the cluster node,

$$MAC_{BS-CN_i} = h_{K_{CN_i}}(ID_{BS}, N_{CN_i}) \tag{66}$$

Upon receiving the message authentication message MAC_{BS-CN_i}, the cluster node computes the message authentication code MAC'_{BS-CN_i} and checks whether or not it is equal to MAC_{BS-CN_i},

$$MAC'_{BS-CN_i} = h_{K_{CN_i}}(ID_{BS}, N_{CN_i}) \tag{67}$$

$$MAC'_{BS-CN_i} \stackrel{?}{=} MAC_{BS-CN_i} \tag{68}$$

After authentication, the BS selects a new nonce N'_{CN_i} and sends it to the cluster node N_{CN_i}; the cluster node N_{CN_i} updates the nonce after sending the monitoring data back to the BS. Because the nonce N_{CN_i} and the session key K_{CN_i} are updated in each session, our scheme achieves the mutual authentication between cluster node and BS.

4.4. Dynamic Key Management

In the dynamic key management phase, the session keys of the sensor and cluster nodes are generated as follows:

$$K'_{SN_i} = h(K_{SN_i}, a_i^1), \quad (\text{where } K_{SN_i} = a_i^2) \tag{69}$$

$$K'_{CN_i} = h(K_{CN_i}, b_i^1), \quad (\text{where } K_{CN_i} = b_i^2) \tag{70}$$

As we mentioned, the next parameters of the hash seed (K_{SN_i}, a_i^1) and (K_{CN_i}, b_i^1) are updated in each session.

Our scheme can solve the replay problem in random pairwise keys pre-distribution. It only needs two pre-stored parameters to produce the session key by using the hash function. When the information transmission is finished, the cluster and the sensor nodes should update the session key and prevent the replay attack after each session.

4.5. Discussion

The proposed scheme supports mutual authentication and fully dynamic key agreement. It is worth mentioning that it embeds the polynomial key function and the GPS location capability in cluster nodes. Each cluster node can leverage on GPS information to find out its best path to the BS, and the polynomial key function can be easily used to create the cluster node session keys $K_{CN_i-CN_j}$ necessary for encrypted communications between adjacent cluster node pairs (*i, j*) along this path. If any of these cluster node gets corrupted, the other ones can use the broadcast message msg_{start} to find out their new best path to the BS. As Table 1 shows that our scheme is superior to other related works.

Table 1. Comparison of the proposed scheme with the most significant related works

Protocol	Our scheme	KMTD[8]	IKDM[7]
Captured attack analysis	Yes	Yes	Yes
Detail security analysis	Yes	Yes	Partial (captured attack)
Stored cost (Cluster node)	One session key, two polynomial function	Two session keys, one base station ID	One session key, two polynomial function
Stored cost (Sensor node)	Two session keys, one cluster node ID	Two session keys, one cluster node ID	Two session keys, one base station ID
Sensor network model	Hierarchical	Hierarchical	Hierarchical
Sensor's homogeneity	Hierarchical	Hierarchical	Homogeneous
Mutual authentication	Yes	Yes	N/A
Dynamic key agreement	Yes	Yes	N/A
GPS capability	Yes (cluster node)	N/A	N/A
Routing protocol	Yes	N/A	N/A

5. Conclusions

In this paper, we proposed a dynamic key management scheme. Our scheme can achieve the following goals:

(1) We provide a dynamic key management to prevent the replay attack.
(2) We use the GPS technology to find the nearest node of the BS to the neighbor cluster nodes.
(3) We proposed a nonce-based mechanism to complete the mutual authentication between the BS and cluster nodes. It can enhance the information security.
(4) We coped with the storage and energy consumption limitations and reduced the computation cost of the sensors.

In the future, we envision that our scheme could be extended to apply polynomial key techniques in different WSNs for more efficient transmission combined with energy control mechanism or for implementing alternate key management strategies.

6. References

1. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Transactions on Information Theory, Vol. 22, No. 6, 644-654. (1976)
2. Rivest, R.L., Shamir, A. Adleman, L.: A Method for Obtaining Digital Signatures and Public-key Cryptosystems. Communications of the ACM, Vol.21, No. 2, 120-126. (1978)
3. Xia, F., Yang, X., Liu, H., Zhang, D., Zhao, W.: Energy-efficient Opportunistic Localization with Indoor Wireless Sensor Networks. Computer Science and Information Systems, Vol. 8, No. 4, 973-990. (2011)
4. Martin, K.M., Paterson, M.: An Application-Oriented Framework for Wireless Sensor Network Key Establishment. Electronic Notes in Theoretical Computer Science, Vol. 192, No. 2, 31–41. (2008)
5. Chan, H., Perrig, A., Song, D.: Random Key Predistribution Schemes for Sensor Networks. 03 Proceedings of the 2003 IEEE Symposium on Security and Privacy, IEEE Computer Society Washington, DC, 11-14 May,2003, USA. 197-213. (2003)
6. Sheu, J.P., Cheng, J.C.: Pair-wise Path Key Establishment in Wireless Sensor Networks. Computer Communications, Vol. 30, No. 11-12, 2365-2374. (2007)
7. Cheng, Y., Agrawal, D.P.: An Improved Key Distribution Mechanism for Large-Scale Hierarchical Wireless Sensor Networks. Ad Hoc Networks, Vol. 5, No. 1, 35–48. (2007)
8. Chen, C.L., Tsai, Y.T., Shih, T.F.: A Novel Key Management of Two-tier Dissemination for Wireless Sensor Network. 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous

Computing (IMIS 2012), Palermo, Italy. 576-579. (2012)
9. Qian, Q., Shen, X., Chen, H.: An Improved Node Localization Algorithm based on DV-Hop for Wireless Sensor Networks. Computer Science and Information Systems, Vol. 8, No. 4, 953-972. (2011)
10. Wang, X., Ma, J., Wang, S., Bi, D.: Distributed Energy Optimization for Target Tracking in Wireless Sensor Networks. IEEE Transactions on Mobile Computing, Vol. 9, No. 1, 73-86. (2010)
11. Blundo, C., De Santis, A., Herzberg, A., Kutten, S., Vaccaro, U., Yung, M.: Perfectly-Secure Key Distribution for Dynamic Conferences. Journal Information and Computation, Vol. 146, No. 1, 1-23. (1998)
12. Palmieri, F.; Ricciardi, S.; Fiore, U.: Evaluating Network-Based DoS Attacks under the Energy Consumption Perspective: New Security Issues in the Coming Green ICT Area," International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), 2011 pp.374-379. (2011)

Chin Ling Chen, PhD, is a member of the Chinese Association for Information Security. From 1979 to 2005, he was a senior engineer at the Chunghwa Telecom Co., Ltd. He is currently a professor of the Department of Computer Science and Information Engineering at Chaoyang University of Technology, Taiwan. His research interests include cryptography, network security and electronic commerce. Dr. Chen had published over 50 SCI/SSCI articles on the above research fields in international journals.

Yu Ting Tsai, born in 1988. Currently he is pursuing his master's degree at the Department of Computer Science and Information, Chaoyang University of Technology. His research interests include WSN and cryptography.

Aniello Castiglione, PhD, joined the Computer Science department ``R. M. Capocelli'' of University of Salerno, Italy, in 2006. He is an active member of IEEE, ACM and IISFA (International Information System Forensics Association). His research interests include Communication Networks, Digital Forensics, Security and Privacy, Security Standards and Cryptography.

Francesco Palmieri, PhD, is an assistant professor at the Engineering Faculty of the Second University of Napoli, Italy. His research interests concern high performance networking protocols and architectures, routing algorithms and network security. He has been closely involved with the development of the Internet in Italy as a senior member of the Technical-Scientific Advisory Committee and of the CSIRT of the Italian NREN GARR.

Received: September 07, 2012; Accepted: March 05, 2013