Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 740392, 13 pages
http://dx.doi.org/10.1155/2013/740392

Research Article

Integrated Protocols to Ensure Security Services in Wireless Sensor Networks

Mohammed Faisal, Jalal Al-Muhtadi, and Abdullah Al-Dhelaan

Department of Computer Science, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia

Correspondence should be addressed to Mohammed Faisal; mfaisal@ksu.edu.sa

Received 19 January 2013; Revised 31 March 2013; Accepted 9 April 2013

Academic Editor: Muhammad Khurram Khan

Security in current/next generation wireless sensor network (WSN) is challenging, due to its special characteristics and the scarcity of energy, and processing power. Thus, many security methods are trying to solve the problem of security in wireless sensor networks. Many of these methods use symmetric cryptography, and others use asymmetric cryptography. Security in WSN demands authentication, confidentiality, integrity, balancing the energy cost, nonreputation, and scalability services. In this paper, we introduce two protocols to address these main security services. The proposed protocols integrate public key and symmetric key algorithms to ensure optimal usage of sensors' energy and processing power to provide adequate security in next generation WSN. The proposed protocols utilize Elliptic Curve Cryptography (ECC), Advanced Encryption Standard (AES), Certification Authority (CA), and Registration Authority (RA) to achieve a reasonable tradeoff.

1. Introduction

WSNs are infrastructureless and fully distributed systems of self-configurable and self-organizing. WSNs are used in industry process control, military sensing and tracking, environmental monitoring, patient monitoring, and so forth [1]. In the last two decades, many research efforts proposed various security services for WSNs [2–6]. These works used symmetric or asymmetric cryptography along with other tools to handle security needs for WSNs [7, 8]. In this paper, we integrate the symmetric and asymmetric key cryptography to ensure security and optimize energy usage. The proposed protocols focus on creating a Light Public Key Infrastructure protocol and Special Cluster based Multipath Routing Protocol. The proposed protocols are Light Public Key Infrastructure (L-PKI) protocol and Secure and Energy-efficient Cluster based Multipath Routing (SECMRP) protocol.

Our main contribution is integrating a lightweight PKI that utilizes ECC and AES crypto algorithms with a secure data transmission service via multipath in an efficient manner, in terms of resource and power consumption. The proposed protocols enhance the security of WSN by providing mutual authentication between neighbor nodes and sending the data via multipath for redundancy.

This paper is organized as follows. Section 2 talks about related work. Section 3 explains the WSN architecture. Section 4 gives an overview of the proposed protocols. Section 5 presents the L-PKI protocol. Section 6 presents the SECMRP protocol. Section 7 describes system's experimentation and results. Section 8 compares results between our work and related work. Finally, Section 9 concludes.

2. Related Work

This work combines key management and route discovery protocols in WSN. Thus, related work can be divided into two major subsections: key management protocols and route discovery protocols.

2.1. Key Management Protocols. Using public key cryptography, we can ensure confidentiality, authentication, and integrity. A PKI is an efficient tool for achieving key management in networks [9]. PKI is greedy in terms of resources. It typically consumes a lot of energy and bandwidth through extensive use of public key crypto. Thus, using PKI in WSN is challenging. Many studies focuses on developing public key cryptographic algorithms that are more efficient energy efficient such as Elliptic Curve Cryptography (ECC) [10]. Several

research groups have successfully implemented public key cryptography in WSNs [4–6].

TinyPK attempts to secure sensor networks with public-key cryptography and implements a public key-based protocol that allows authentication and key exchange between an external party and a sensor network [11]. TinyPK focuses on supporting confidentiality and source authentication for sensor network traffic. TinyPK is based on RSA 1024 bits [12]. TinyPK is implemented on UC Berkeley MICA2 motes using the TinyOS development environment. TinyPK uses other symmetric encryption and Diffie-Hellman key agreement algorithm [13] to exchange secret keys. TinyPK uses a CA whose public key has to be preloaded into the nodes during preconfiguration phase. In our approach we use ECC instead of RSA, which would give an advantage for performance and battery consumption. Also, we avoid overusing Diffie-Hellman key exchange to avoid potential man-in-the-middle attacks.

μPKI [9] is a lightweight implementation of PKI for WSNs. In this protocol, two handshakes are used: the first handshake between the BS and each sensor in the network and the second handshake between each pair of node in the network intended to secure sensor to sensor communication. In μPKI, only the BS needs to be authenticated. μPKI uses the public key cryptography in the key distribution operation. μPKI assumes the existence of an offline communication protocol to distribute the public key of the BS to each sensor in the network. Thus, μPKI uses the public key in the handshake between the BS and sensors. In this handshake, each sensor generates and encrypts the session key using the public key of the BS. The purpose of the first handshake is to create a secure end-to-end transmission between each node and the BS. The second handshake, which is between each pair of node, is used to establish a secure channel between them. In this handshake, one of these nodes sends request to the BS. This request contains the identifier of the corresponding node. The BS responds to this request by generating a random key then encrypts a copy for each sensor using the corresponding session key, which has been created in the first handshake. Actually, in μPKI protocol the security can be broken by knowing the public key of the BS. Any malicious node can then encrypt a session key using the BS's public key. The BS plays the main role in the operation of creating the session key between each pair of nodes. Thus, a lot of traffic exists.

2.2. Route Discovery Protocols. Routing in WSNs is challenging, as there are many routing protocols: some producing a single path only, while others produce multipath. To ensure the security of WSNs, the scheme should ensure the security of route discovery and data transmission. Several protocols have been proposed for route discovery.

Secure and Energy-Efficient Multipath Routing Protocol (SEEM) [14] proposed multipath routing protocols where instead of using the initial lowest energy route for communication, the BS finds multipath to the source of the data and selects one to use during the communication. Furthermore, the BS updates the available energy of each node along the path depending on the amount of packets being sent and received [14]. The BS then uses the updated energy state of each node to select new paths. BS in SEEM works as a server, which floods the query to the network. The node that satisfies the query will send a request to the BS for sending a path. SEEM does not use any cryptographic mechanisms to address confidentiality. SEEM constructs disjoint and braided paths using a modification of the Breadth First Search algorithm [14].

INtrusion-tolerant routing protocol for wireless SEnsor NetworkS (INSENS) proposes a multipath routing protocol that minimizes the computation, communication, storage, and bandwidth required at the sensor nodes in the operation of route discovery [15]. INSENS does not rely on detecting intrusions but rather tolerates intrusions by bypassing the malicious nodes. An important property of INSENS is that while a malicious node may be able to compromise a small number of nodes in its vicinity, it cannot cause widespread damage in the network [15]. In INSENS, the BS plays an important role in the operation of routing discovery. The BS always receives knowledge of the topology of the network and finds multipath to each node in the network, and then the BS unicasts the multipath table to each corresponding node [15].

SECMRP uses the concept of multipath and clustering to deal with security and efficiency. SECMRP enhances the security issue of SEEM and, at the same time, uses two disjointed paths to send sensed data to the CH. In SEEM, the BS works as a server, which floods the query to the network, and the node which satisfies the query will send a request to the BS for a path, but CH in SECMRP periodically sends paths to each node. SEEM increases the network lifetime about 35% as compared to directed diffusion. So that we can say that SECMRP is energy efficient and is able to enhance network lifetime due to the roles of CH.

Early preliminary motivation and design for this work were published in [8]. In this paper, we revise the design, provide full implementation, and compare results with related work.

3. The Proposed WSN Architecture

In order to ensure the security services, we divide the sensor network into many clusters and assume the existence of three types of nodes (Figure 1 illustrates the proposed architecture):

(1) sensing nodes;

(2) cluster heads (CH) (one or more);

(3) base stations (BS) (One or more).

Table 1 includes the tasks of WSNs components.

4. Proposed Protocols

The proposed protocols are designed to ensure authentication, confidentiality, balancing the energy cost, nonrepudiation, scalability, and integrity in WSNS. This is done by proposing two protocols, Light Public Key Infrastructure L-PKI protocol and Secure and Energy-efficient Cluster based Multipath Routing SECMRP.

Table 1: Tasks of WSN components.

Member	Explanation
Base station	(i) Collects and processes information from CH. (ii) Assigns identifier ID to both CHs and ordinary nodes. (iii) Determines the CH of each node in the network by assigning the CH's ID and public key to each node, so each node knows its cluster, and each cluster knows its nodes.
CH	CA roles (i) Issues and delivers the digital certificate of both CHs and ordinary nodes "offline." (ii) Signs the digital certificate "offline." RA roles Binds the digital certificates and ID of both CHs and ordinary nodes. Other roles (i) Ensures the security of proposed WSNs via (a) using its public key for authentication operation of each node and its neighbors (in all nodes that belong to the same cluster), (b) validating the certification of node if the node belongs to other cluster via cross-authentication. (ii) Aggregates the sensed information. (iii) Sends the sensed information to the BS. (iv) Acts as a gateway between the sensed region and the BS.
Ordinary nodes	Sense or monitor (according to the application requirement) information and send it to the BS via secure disjoint multipath through the CHs.

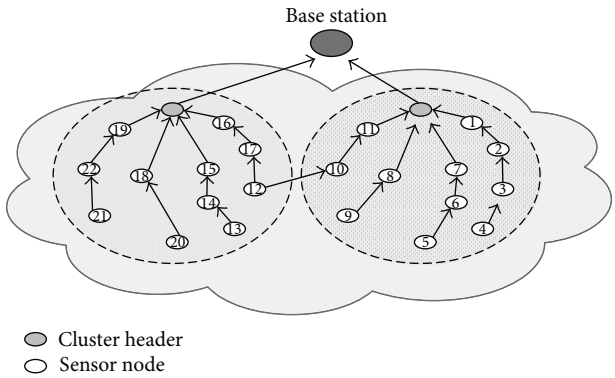

Figure 1: The proposed architecture.

The proposed protocols (L-PKI and SECMRP) prevent from several types of attacks, internal, passive, and impersonation (spoofing) attacks. Also, modification of protocol messages and routing table overflow attacks are addressed in the proposed protocols.

5. L-PKI Protocol

L-PKI proposes a lightweight implementation of PKI, suitable for WSNs. L-PKI proposes a protocol that utilizes public key cryptography only in authentication and in generating a session key between the cluster head (CH) and each sensor node in the cluster. L-PKI sets up secured links between each node and its neighbors, each node and its CH, and each CH and the BS. These secured links are used to transmit data to the BS via CHs.

For L-PKI, we assume the following:

(1) CHs have more computational and energy power compared to other sensor nodes, which allow them to work as a limited Certification Authority (CA) and Registration Authority (RA);

(2) the BS and CHs are trusted entities;

(3) each sensor node has the capability to use symmetric encryption and Elliptic Curve Cryptography (ECC) operations (hardware or software).

It is established that ECC is a relatively efficient public key crypto, relevant to RSA and other popular algorithms, thus it is suitable for WSNs [4–6, 16, 17]. L-PKI utilizes ECC with keys of 160 bits [18]. There are many symbols used in this protocol. To simplify the description of the L-PKI protocol, Table 2 explains the meaning of the symbols used in the protocol description. L-PKI is composed of two phases as follows:

(1) initial authentication and Key establishment phase, which contains

(a) predeployment step,

(b) authentication and key establishment step,

(2) addition of new node phase.

5.1. Initial Authentication and Key Establishment Phase

5.1.1. Predeployment Step. In order to increase the live time of the WSNs, L-PKI moves most of the load of the PKI operations to the CHs. According to the proposed protocol, the CHs will work as a lightweight PKI. Predeployment step includes the following:

(1) each CH and node generate its private and public keys (CHK_{pub}, CHK_{prv}), (NK_{pub}, NK_{prv});

(2) BS generates a unique identifier, CH-ID (8 bits) to each CH in the network. 8 bits CH ID can have up to 256 clusters;

Table 2: Output of LPKI.

Symbols	Description	Size
CH-ID	Identify definition of the CH	8 bits
N-ID	Identify definition of sensor node (N can be any letter according to the name of node)	8 bits
CHK_{pub}/CHK_{prv}	The private and public keys of CH	160 bits
NK_{pub}/NK_{prv}	The private and public keys of sensor nodes (N can be any letter according to the name of node)	160 bits
CH_cer	Digital certificate of CH	192 bits
N_cer	Digital certificate of sensor node (N can be any letter according to the name of node)	192 bits
CH_sig	The signature of CH (N can be any letter according to the name of node)	320 bits
N_sig	The signature of node (N can be any letter according to the name of node)	320 bits
SH_XYK	AES shared key between neighbors node (X and Y can be any letter according to the name of nodes)	128 bits
NBR_LIST	List contains all neighbors of each node	

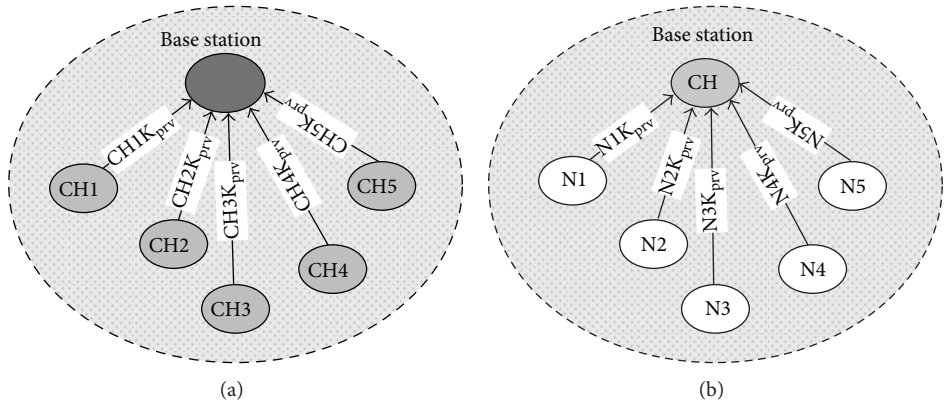

Figure 2: (a) Delivering CHs' key and (b) delivering nodes' key.

(3) BS generates a unique ID (N-ID) (8 bits) to each sensor node in the network;

(4) each CH delivers its public key to BS, as illustrated in Figure 2(a);

(5) each node delivers its public key to its CH. As illustrated in Figure 2(b);

(6) BS issues digital certificates for each CH (CH_cer) and signs it;

(7) BS delivers the digital certificates to each corresponding CH;

(8) CHs issue digital certificates for each sensor nodes (N_cer) and sign it. Figure 3(a) illustrates this step;

(9) CHs deliver the digital certificates to each corresponding node. Figure 3(b) illustrates this step.

L-PKI utilizes a compressed certificate format, which only keep needed fields, (mainly, nodes or CHs IDs, timestamp, and signature). L-PKI uses these fields in order to ensure the security with lowest cost as follows:

(a) CH_cer = CHK_{pub} "160 bits" | CH-ID "8 bits" | Timestamp "16 bits" | BS-ID "8 bits" = 192 bits,

(b) N_cer = NK_{pub} "160 bits" | N-ID "8 bits", Timestamp "16 bits" | CH-ID "8 bits" = 192 bits.

We recommend using Elliptic Curve Digital Signature Algorithm (ECDSA-160) as a signature algorithm [16, 18], so that the size of the signature will be 320 bits, because we use ECC-160. Figure 4 illustrates thesignature operation.

After applying the predeployment step, each node has its own digital certificate N_cer, public key NK_{pub}, and private key NK_{prv} and also knows the public key (CHK_{pub}) and the CH-ID of its CH.

5.1.2. Authentication and Key Establishment Step. At this step, each node in the same cluster must authenticate each other. Figure 5 illustrates the authenticate operation. After the authentication step, each node will trust its neighbor. After the predeployment step, each node will have its own digital certificate N_cer, public key NK_{pub}, private key NK_{prv}, the public key (CHK_{pub}) of its CH, and the CH-ID of its CH. Each node must do the following steps in order to authenticate with its neighbors:

(1) node "A" sends its digital certificate A_cer to its neighbor "B";

(2) node B receives the certificate of node A "A_cer" and verifies it by using the public key of its CH CHK_{pub} as follows:

(i) Size of message = 192 bits "A_cer" = 192 bits;

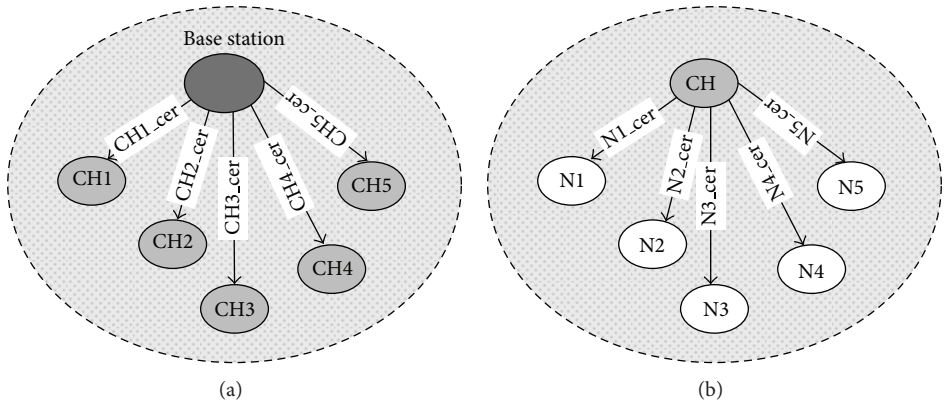

Figure 3: (a) Delivering CHs' digital and (b) delivering nodes' digital certificates.

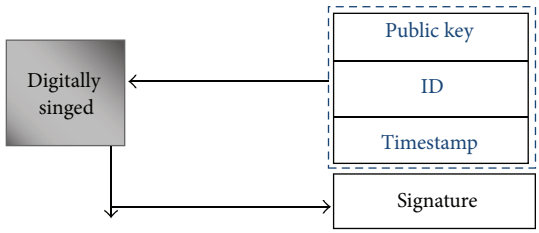

Figure 4: Signature operation.

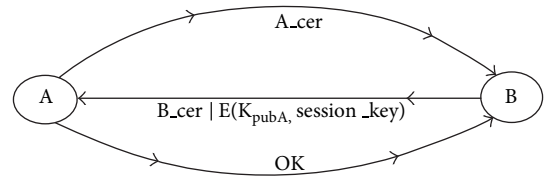

Figure 5: Authentication operation.

(3) at node B, if A_cer is valid, node B creates and encrypts a session key SH_ABK. This key is encrypted by using the public key of node A "AK_{pub}" as follows:

(i) L-PKI uses AES-128, and thus the key size is 128 bits;

(ii) size of message = key 128 bits "SH_ABK" + 192 bits "A_cer" + 128 bits MAC = 448 bits. Figure 5 illustrates the authentication operation;

(4) node B sends its digital certificate B_cer with its session key "SH_ABK" and MAC;

(5) node A receives the certificate of node B "B_cer" and verifies it by using the public key of the CH CH_{Kpub};

(6) if B_cer is valid, node A decrypts SH_ABK by using its private key AK_{prv};

(7) create MAC and compare it with the received MAC;

(8) node A responds by sending OK message with timestamp to node B. The OK message is encrypted by the session key SH_ABK. This message will act as a challenge message between nodes A and B;

(9) if both nodes A and B successfully validate the certificate,

(i) node A adds node B to its neighbor list,

(ii) node B adds node A to its neighbor list.

Now, each node trusts its neighbor node.

5.2. Adding New Node Phase. In case of a new node joining the network, it must be prepared by the BS and CH; that is, it must follow the predeployment step and authentication and key establishment step. At first, the new node has to create NK_{pub}, NK_{prv} and loads the public key of the CH. The BS must assign CH-ID and assign node identifier N-ID. Then, the node must communicate with its CH to prepare its certificate N_cer and determine the cluster of the node. After this, the new node is deployed to the network. The location of the new node is not relevant here, because it can authenticate with any node even a node from another cluster. In the authentication step, if the node belongs to another cluster, the other node sends the node's certificate to the CH. Thus, the CH verifies the certificate by using the public key of the CH which the node belongs to. Figure 6 shows this situation. After the authentication, the new node can establish a new session key.

6. SECMRP Protocol

SECMRP enhances the security issue of SEEM [14] and at the same time uses disjointed paths to send data to the CH. In SEEM, the BS works as a server, which floods the query to the network, and the node which satisfies the query will send a request to the BS for a path. In SECMRP, each CH collects a neighbor table of each node in its cluster then creates and delivers two disjointed paths from CH to each node. SECMRP assumes that each node knows its neighbors, has a shared key with each neighbor and a shared key with its CH. This is done after applying L-PKI protocol. After L-PKI, each node has a unique ID N_ID, a digital certificate N_cer signed by its CH, list of all authenticated neighbors NBR_LIST, public keys of CH CHK_{pub}, and shared key with each

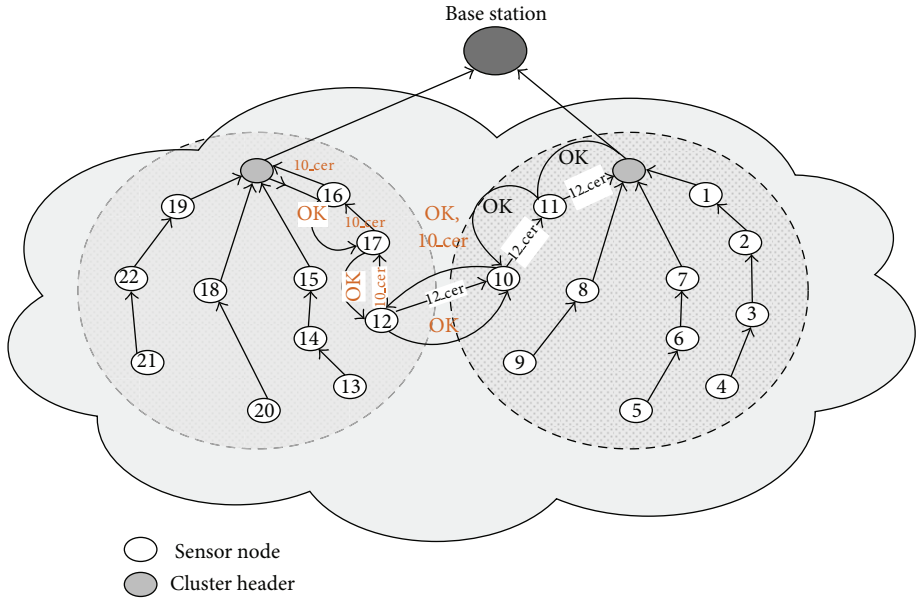

Figure 6: Adding a new node.

Table 3: Symbols of SECMRP.

Symbols	Description	Bits
CH-ID	Identify definition of the CH	8
N-ID	Identify definition of sensor node (N can be any letter according to the name of node)	8
CHK_{pub}/CHK_{prv}	The private and public keys of CHs	160
NK_{pub}/NK_{prv}	The private and public keys of sensor nodes (N can be any letter according to the name of node)	160
CH_cer	Digital certificate of CH	192
N_cer	Digital certificate of sensor node (N can be any letter according to the name of node)	192
CH_sig	The signature of CH (N can be any letter according to the name of node)	320
N_sig	The signature of node (N can be any letter according to the name of node)	320
SH_XYK	AES shared key with each neighbor node (X and Y can be any letter according to the name of nodes)	128
SH_NCHK	Shared key with the CH (N can be any letter according to the name of node)	128
Received_pkts	Table contains the receiving packet sequence number in each node	
P_SEQ_NUM	Packet sequence number	8
N_MAC_val	MAC values of node N (N can be any letter according to the name of node)	128 bits
A_*NBR*	The neighbors of node A	Varies (32 per neighbor)
NBR_SEN	Packet from the CH to ask each node to send its neighbors	160 bits
NBR_LIST	Packet contains all neighbors of node	$424 + n * 32\ (n = 4) \rightarrow = 552$
DATA_PACKET		320 + Size (data)

neighbor node SH_XYK. Table 3 describes all symbols used in SECMRP.

SECMRP has three phases *secure route discovery*, *secure data transmission*, and *route maintenance*. Secure route discovery phase is responsible for finding secure disjoint multipath or partially disjoint multipath between each node and its CH. Data transmission phase is responsible for transmitting the sensing data. At route maintenance phase, each CH updates available energy of each node, which is participating in the communication and according to the available energy on each node reselects a new path to the source node.

6.1. Secure Route Discovery. Secure route discovery phase starts by applying the L-PKI. Secure route discovery has three steps: NBR_LIST requesting, NBR_LIST sending, and NBR_LIST receiving and paths creation steps.

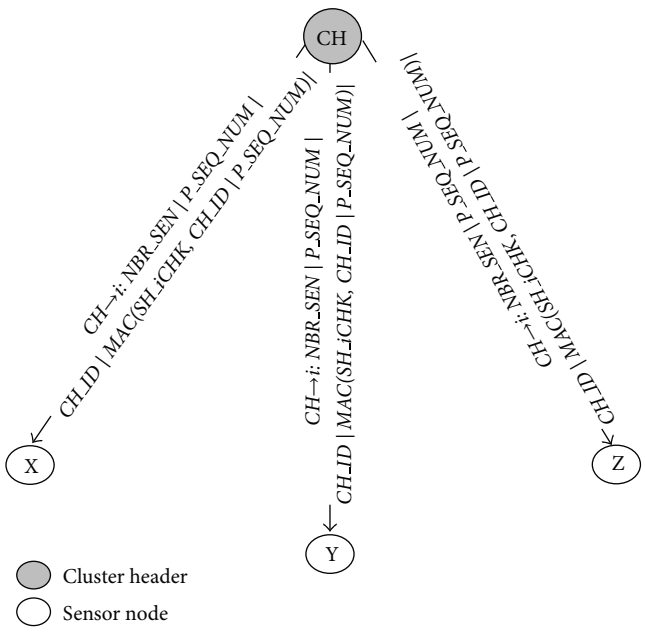

Figure 7: CH requests neighbor list of each node.

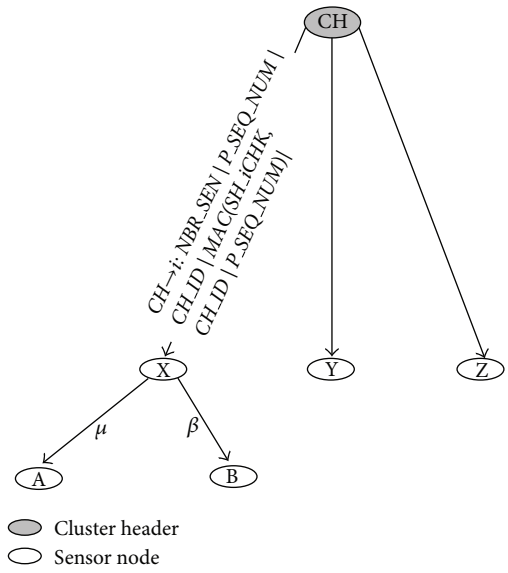

Figure 8: CH requests neighbor list of each node.

```
packet NBR_SEN_Packet
        {
string Packet_Type;
string P_SEQ_NUM;
int CHID;
string Packet_MAC;
        };
```

Data Structure 1: NBR_SEN data type.

6.1.1. NBR_LIST Requesting. Each CH of each cluster starts the routing discovery operation by sending NBR_SEN "neighbor sending" packet to each node on its cluster. The size of NBR_SEN packet is 160 bits and contains packet sequence number "*P_SEQ_NUM*" CH ID "*CH_ID*", and MAC values of the *CH_ID* and *P_SEQ_NUM* "*CH_MAC_val*" the following format (as illustrated in Figure 7.)

for all i where i is a neighbor of CH
CH → *i: NBR_SEN* | *P_SEQ_NUM* | *CH_ID* | *MAC* (*SH_CHiK, CH_ID* | *P_SEQ_NUM*).

Each node receiving NBR_SEN packet does the following:

(1) checks if the node has received this NBR_SEN by searching *P_SEQ_NUM* in the table received_pkts. If the packet received once, then drops this packet and does not rebroadcast it. Otherwise stores *P_SEQ_NUM* in received_pkts table;

(2) computes the MAC value of *CH_ID* and *P_SEQ_NUM* by using the shared key SH_CHiK MAC (SH_CH*i*K, *CH_ID* | *P_SEQ_NUM*) and compares it with the MAC value *CH_MAC_val* of the NBR_SEN packet. If the MAC values are equal move to step 4;

(3) computes the MAC value of CH_ID and *P_SEQ_NUM* by using the shared key of each neighbor node,

for all j where j is a neighbor of i,
MAC (*SH_ijK,* CH_ID | *P_SEQ_NUM*);

(4) rebroadcasts the NBR_SEN packet to its neighbors,

for all j where j is a neighbor of i
i → *j: NBR_SEN* | *P_SEQ_NUM* | *CH_ID* | *MAC* (*SH_ijK, CH_ID* | *P_SEQ_NUM*).

The first step ensures that no NBR_LIST packet is broadcasted more than one time for each node. The second step ensures the integrity and avoids replay attaches. Thus the communication overheads for transmitting and receiving packets are reduced. The following packets and Figure 8 show this operation (X is the current node, and A, B are its neighbors) as follows:

X → ***A***: *NBR_SEN* | *P_SEQ_NUM* | *CH_ID* |
MAC (**SH_XAK**, *CH_ID* | *P_SEQ_NUM*) *(μ).*

X → ***B***: *NBR_SEN* | *P_SE Q_NUM* | *CH_ID* |
MAC (**SH_XBK**, *CH_ID* | *P_SEQ_NUM*) *(β).*

The format "data type" of the NBR_SEN packet is shown in Data Structure 1. Each node which has received the NBR_SEN packet responds by sending a neighbor list (NBR_LIST) packet to the CH.

6.1.2. NBR_LIST Sending. After each node has received the NBR_SEN packet and completed all operations that belong to the NBR_SEN packet, it becomes able to send the NBR_LIST packet. Each node prepares the NBR_LIST packet and sends it directly to the node, which has already sent the NBR_SEN packet as follows:

for all i where i is any node in the cluster, j is the node, which has sent the NBR_SEN packet to the node i.

```
packet NBR_LIST_Packet
  {
string Packet_Type;
string P_SEQ_NUM;
int Node_ID;
string Node_Name;
string Packet_MAC_Neighbors; //Mac (SH_key, P_SEQ_NUM|ID)
string Encrypted_Session_Key; //use to save the E (CHK_pub, Session_key)
int Power_State;
string Encrypted_Neighbor_List; // E (Session_key, NBR_LST)
string Packet_MAC_Src_CH; //Mac (SH_ACH, P_SEQ_NUM | ID | E (CHK_pub, Session_key)
| Power_State | Encrypted_Neighbor_List)
  };
```

Data Structure 2: NBR_LIST packet data type.

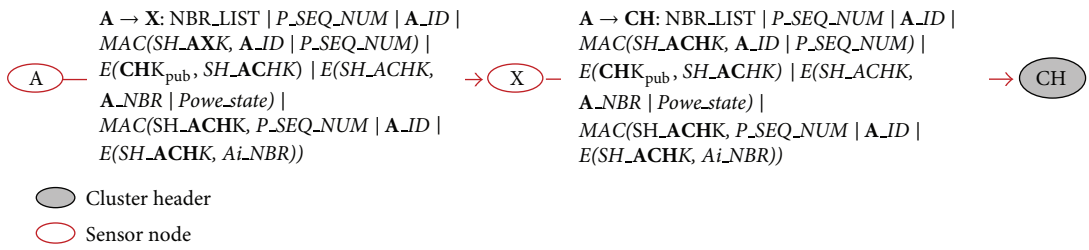

Figure 9: NBR_LIST packet.

$i \rightarrow j$: *NBR_LIST* | *P_SEQ_NUM* | *i_ID* | *MAC* (*SH_ijK, i_ID* | *P_SEQ_NUM*) | *E* (CHK$_{pub}$, *SH_iCHK*) | *E* (*SH_iCHK, i_NBR* | *Powe_state*) | *MAC* (SH_*ij*K, *P_SEQ_NUM* | *i_ID* | *E* (*SH_iCHK, i_NBR*)).

Each node has its neighbor list as a result from applying the L-PKI protocol. Each node receiving NBR_LIST packet does the following (*i* is the previous node, and *j* is the current node):

(1) check if node has received this NBR_LIST by searching *P_SEQ_NUM* in the table received_pkts; if the packet has been received once, then drop this packet; Otherwise store *P_SEQ_NUM* in received_pkts table of the node;

(2) compute the MAC value of *i*_ID and *P_SEQ_NUM* by using the shared key SH_*ij*K MAC (SH_*ij*K, *i*_ID | *P_SEQ_NUM*) and comparing it with the MAC value of the NBR_LIST; if equal, it moves to the next step; otherwise drop the packet;

(3) compute the MAC value of *j_ID* and *P_SEQ_NUM* by using the shared key between the *j* node and the node, which sends the *NBR_SEN* packet;

(4) send the *NBR_LIST* packet to node, which sends the *NBR_SEN* packet, until the packet reach the CH;

(5) if the current node is the CH, then decrypt "*E* (CHK$_{pub}$, *SH_iCHK*)" by using its private key, check the integrity by computing the MAC of *MAC* (*SH_iCHK, P_SEQ_NUM* | *i_ID* | *E* (*SH_iCHK, i_NBR*)), and compare it with the MAC value of the NBR_LIST packet. If the MAC values are equal, then CH decrypts neighbor information using the session key between the sender node and the CH; otherwise drop the packet. The format "data type" of the NBR_LIST packet is illustrated in Data Structure 2. The following packets and Figure 9 explain this operation:

A $\rightarrow$ ***X*** $\rightarrow$ ***CH***,

A $\rightarrow$ ***X***: NBR_LIST | *P_SEQ_NUM* | ***A***_*ID* | *MAC* (*SH_**AX**K, **A**_ID* | *P_SEQ_NUM*) | *E* (**CH**K$_{pub}$, *SH_**AC**HK*) | *E* (*SH_ACHK, **A**_NBR* | *Powe_state*) | *MAC* (SH_**ACH**K, *P_SEQ_NUM* | ***A***_*ID* | *E* (*SH_**ACH**K, Ai_NBR*)),

A $\rightarrow$ ***CH***: NBR_LIST | *P_SEQ_NUM* | ***A***_*ID* | *MAC* (*SH_**ACH**K, **A**_ID* | *P_SEQ_NUM*) | *E* (**CH**K$_{pub}$, *SH_**AC**HK*) | *E* (*SH_ACHK, **A**_NBR* | *Powe_state*) | *MAC* (SH_**ACH**K, *P_SEQ_NUM* | ***A***_*ID* | *E* (*SH_**ACH**K, Ai_NBR*)).

6.1.3. NBR_LIST Receiving. As we have said, if the current node is the CH, then decrypt "*E* (CHK$_{pub}$, *SH_iCHK*)" by using its private key, check the integrity by computing the MAC of *MAC* (*SH_iCHK, P_SEQ_NUM* | *i_ID* | *E* (*SH_iCHK, i_NBR*)), and compare it with the MAC value of the NBR_LIST

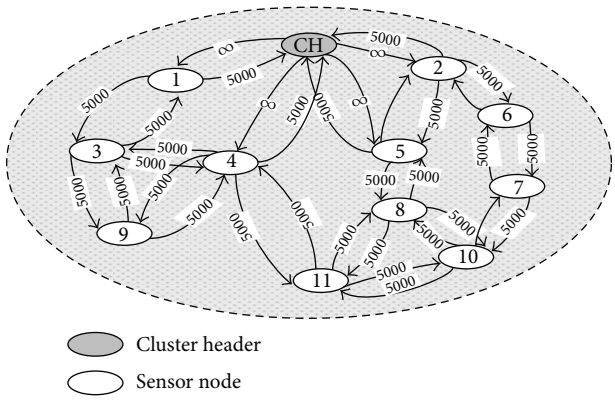

Figure 10: The initial weighted directed graph of one cluster.

packet. If the AMC values are equal, then CH decrypts neighbor information using the session key between the sender node and the CH; otherwise, drop the packet. After each CH has received neighbors' information and the state of power of all nodes on the cluster, each CH has a "vision" of the topology of the cluster nodes. Thus, CH is able to find secure disjoint multipath or partially disjoint multipath between each node and CH and select one of them according to specific algorithm.

6.1.4. Finding Secure Disjoint/Partially Disjoint Multipath. According to neighbors information each CH constructs a weighted directed graph and finds the multipath from the CH to every source node. SECMRP finds disjoint/partially disjoint multipath and selects N of them according to the minimum hops or the maximum energy available for each node on the path. Figure 10 shows a weighted directed graph G (V, E) of one cluster in the proposed WSNs. The weight of an edge in the corresponding graph of the network represents the available energy on the source (head) node. Due to the high energy of the CH, its weight is infinite. We assume that all other edges have the same weights 5000 mJ (the power state, which was sent from each node) after the CH has received NBR_LIST packets of all nodes.

As mentioned before, when the CHs receive the NBR_LIST packets, they start to calculate the shortest path among the available multipath to each node. The shortest path has the minimum sum of energy consumed for transmission of the packet, which is the path with minimum hops or the path with maximum energy available for each node. SECMRP modifies the Breadth First Search (BFS) algorithm (MBFS), similar to [14], but SEEM selects one path while SECMRP selects two paths. The main points of MBFS algorithm are as follows (assuming we need to transmit data from node 4 to CH, as shown in Figure 11):

(1) whenever it finds "$N = 2$" shortest two paths to node number 4, the MBFS algorithm checks if the weight of each edge on the paths is greater than the predefined level (SECMRP uses five define levels of energy limitation, which is similar to SEEM [14]. Each level is twice of the lower level). If the weight of each edge on the path is greater than the predefined level MBFS returns these paths as the shortest paths. Otherwise go to next step;

(2) if there is any node in the path, whose weight is less than the predefined level, MBFS does not select it and continues the searching until it finds the second shortest path. Thus, MBFS tries to balance the energy cost in the whole cluster;

(3) if MBFS cannot find any shortest path under current level of the energy limitation, it means that each path has at least one node whose energy is less than current energy limitation;

(4) thus MBFS algorithm uses the lower energy limitation and resumes the searching operation with the new level of energy limitation;

(5) if MBFS algorithm cannot find any path under any level of the energy limitation, it means that node 4 is unreachable and the CH cannot get the data from this node.

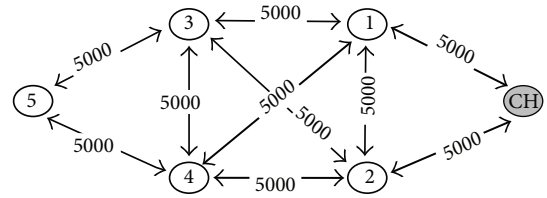

Figure 11: Weighted directed graph of on cluster in the WSN.

Periodically, each node sends its power state to the CH, and CH performs the MFBS for all nodes in the cluster and selects the best two paths between each node and CH then sends them to the corresponding node.

In the previous example, the energy limitation levels are 2500, 1250, 652, 312, and 156. We can realize that the path from source node to the CH goes through different nodes. Therefore, sometimes under specific energy limitation levels, we cannot find a path to specific node, but we can find multipath for other sensor nodes. Thus, each node must have its own energy limitation array. The maintaining and updating operations of this energy limitation array must be independent to each node. This ensures a high lifetime for the sensor network by balancing the energy cost.

6.2. Secure Data Transmission. In the secure route discovery phase each CH is responsible for finding secure disjoint multipath or partially disjoint multipath between each node and its CH. Thus, each node becomes able to transmit its sensed data securely. After the secure route discovery phase and upon the application. CH must have sent the best ($P = 2$) energy efficient paths to each node. Now, nodes become able to send the sensed data to the CH. Secure data transmission phase has two steps:

(1) in the first step, the sensor node encrypts and sends data to the CH;

(2) the second step takes place at the CH; CH aggregates, compresses the received data, which can be from more than one node, and sends the aggregated data to the BS.

By applying the MBFS algorithm, the paths will be "$4 \rightarrow 2 \rightarrow CH$" and "$4 \rightarrow 1 \rightarrow CH$". According to the paths, the source node divides the data to N parts "in this example $p = 2$", and makes the following two packets, and sends them to the nodes number 1 and 2. The data packets "DATA_PACKET" have the following formats:

4 → 1: *DATA_PACKET* | *P_SEQ_NUM* | *4_ID* | *Path* | *MAC* (*SH_41K, P_SEQ_NUM* | *4_ID* | *Path*) | *E* (*SH_4CHK, DATA_PART1*) | *MAC* (*SH_4CHK, P_SEQ_NUM* | *4_ID* | *E* (*SH_4CHK, DATA_PART1*)),

4 → 2: *DATA_PACKET* | *P_SEQ_NUM* | *4_ID* | *Path* | *MAC* (*SH_41K, P_SEQ_NUM* | *4_ID* | *Path*) | *E* (*SH_4CHK, DATA_PART2*) | *MAC* (*SH_4CHK,* | *P_SEQ_NUM* | *4_ID* | *E* (*SH_4CHK, DATA_PART2*)).

To avoid several type of attacks and to balance the power conception in the network, SECMRP divides the data to two parts and sends them via two disjointed paths "$4 \rightarrow 2 \rightarrow CH$ and $4 \rightarrow 1 \rightarrow CH$."

As we can see in the first packet the 4_ID is the ID of the source node. SECMRP uses the *MAC* (*SH_* ***41K***, *P_SEQ_NUM* | *4_ID* | *Path*) to insure the *authentication* between it and node number 1 and to avoid several types of attacks. SECMRP ensures the *confidentiality* by encrypting the *DATA_PART1* by using the shared key of node 1 and CH *E* (*SH_ACHK, DATA_PART1*). To insure the integrity between the source node 1 and the destination "CH," SECMRP uses the MAC function and using the shared key of source node and the CH (*MAC* (*SH_ACHK,* | *P_SEQ_NUM* | ***4***_*ID* | *E* (*SH_ACHK, DATA_PART1*))).

The following section explains the operation of one packet, and the other packet has the same operation. Each intermediate node, which receives the *DATA_PACKET* packet, does the following things:

(1) checks if it has received this *DATA_PACKET* by searching *P_SEQ_NUM* in the received_pkts table; if the packet has been received once, then drop this packet. Otherwise store *P_SEQ_NUM* in received_pkts table and move to next step;

(2) computes the MAC value of *4_ID, P_SEQ_NUM,* and *Path* by using the shared key SH_41K MAC (SH_41K, *4_ID* | *P_SEQ_NUM* | *Path*) and compares it with the MAC value of the DATA_PACKET packet. If they are equal, it moves to next step;

(3) computes the MAC value *4_ID, P_SEQ_NUM.* In the first step, the source sensor node sends the data packet to the CH via the secure path that CH has sent to the source node.

Let us take the example of Figure 11. The initial multipath from the source node 4 to CH is illustrated in Table 4. Assume

Table 4: All available paths from node 4 to CH.

Seq.	Path	Available energy	Energy limitation
1	4 → 1 → CH	5000-5000	2500
2	4 → 2 → CH	5000-5000	2500
3	4 → 2 → 1 → CH	5000-5000-5000	2500
4	4 → 1 → 2 → CH	5000-5000-5000	2500
5	4 → 3 → 2 → CH	5000-5000-5000	2500
6	4 → 3 → 1 → CH	5000-5000-5000	2500

```
packet Secure_Date_Packet
  {
string Packet_Type;
string P_SEQ_NUM;
int Node_ID;
string Path;
string Packet_MAC_Neighbors;
string Encrypted_Data;
string Packet_MAC_Src_CH;
  };
```

Data Structure 3: DATA_PACKET data type.

that the number of paths, which is needed to send data is two $P = 2$ and the levels of energy limitation are 2500, 1250, 652, 312, and 156. And *Path* MAC (SH_1CHK, *4_ID* | *P_SEQ_NUM* | *Path*) by using the shared key between it and next node in the path "*i*", in this example CH "SH_1CHK". And add it to the *DATA_PACKET* packet. The following packet explains this step:

1 → CH: DATA_PACKET | *P_SEQ_NUM* | ***4***_*ID* | *MAC* (*SH_****1CHK***, *P_SEQ_NUM* | *4_ID* | *Path*) | *E* (*SH_4CHK, DATA_PART1*) | *MAC* (*SH_****4CHK***, *P_SEQ_NUM* | ***4***_*ID* | *E* (*SH_4CHK,* ***DATA_PART1***));

(4) sends the new *DATA_PACKET* packet to the next node "*next hop*" in the path;

(5) at the CH, CH performs step 2, 3 and checks the integrity by comparing the MAC values *MAC* (*SH_****4CHK***, *P_SEQ_NUM* | ***4***_*ID* | *E* (*SH_4CHK,* ***DATA_PART1***)), and decrypts *E* (*SH_****4****CHK,* ***DATA_PART1***) using the shared key between source node and CH *D* (*SH_****4****CHK, E* (*SH_****4****CHK,* ***DATA_PART1/2***)).

After the decryption operation, CH sends an acknowledgement (ACK) to the source node. The format "data type" of the DATA_PACKET packet is illustrated in Data Structure 3.

The second step starts after the data becomes available at the CH. CH becomes able to aggregate and compress the received data, which can consist of more than one node. After aggregation and compression operations, the CH encrypts the aggregated data and sends it to BS. The following packet illustrates the aggregated data packet:

CH → BS: *DATA_PACKET* | *P_SEQ_NUM* | ***CH***_*ID* | *MAC*

(*SH_**CHBSK***, *P_SEQ_NUM* | ***CH**_ID* | *Path*) | *E* (*SH_**CHBS**K, DATA*) | *MAC* (*SH_**CHBS**K,*
P_SEQ_NUM | ***CH**_ID* | *E* (*SH_**CHBS**K,*
DATA)).

From the *DATA_PACKET* packet, we can realize that the communication is between the CH and the BS. In this step, SECMRP ensures the authentication and integrity between the CH and BS using MAC and ensure confidentiality by encrypting the *DATA_PACKETE (SH_**CHBS**K, DATA)* using the shared key between CH and the BS.

6.3. Route Maintenance. In SECMRP, each CH has a vision of the topology of its members, and knows the available energy of each node. All this information allows the CH to become able to maintain the route as needed. As discussed, each CH constructs a weighted directed graph and finds the multipath from the CH to every source node by using the MBFS algorithm. The initial information of the power state of each node has been received with the NBR_LIST packet. In the route maintenance phase, each node sends periodically the state of its power to the CH, so each CH can get updated information of power state of each node on the cluster.

SECMRP can maintenance the route by decreasing the weight when the source node sends or receives packets. If the available energy of the node reaches to the energy limitation level, then it uses another path. It means that if one node on the shortest path has energy less than the specific level, then the MBFS discards this path and continues searching for the second shortest path.

7. Experimentation

To validate our proposed protocols, we used OMNeT++4.1 as a simulator, C++ as a programming language, and Crypto++ 5.6.1 as a library for cryptographic algorithms. OMNeT++ provides an integrated development environment based on the eclipse platform and C++ simulation library and framework [20]. OMNeT++ is used for simulating wired and wireless communication networks. Crypto++ 5.6.1 is a C++ library for cryptographic algorithms which implements ECC, AES-128, ECDSA-160 [16, 18, 21], and the other crypto functions needed by the protocols.

7.1. L-PKI Experimentation Results. L-PKI is tested in two different scenarios: the first one was with 10 sensor nodes, two CHs and one BS, and the second scenario was with 5 sensor nodes, one CHs and one BS. In both scenarios, nodes are connected with a channel that has a data rate of 250 Kbps and 100 Mbps for CHs and base stations. Figure 12 illustrates the experimental work of L-PKI protocol.

Energy Cost in L-PKI. We can estimate the energy cost of any key management scheme by calculating the energy required for the execution of cryptographic operation and the energy needed for transmitting the required data. There are many studies, which concentrate on computing the energy cost in WSNs [4, 6, 16, 19]. Each study uses different models. To analyze the energy cost of L-PKI, we can compute the energy

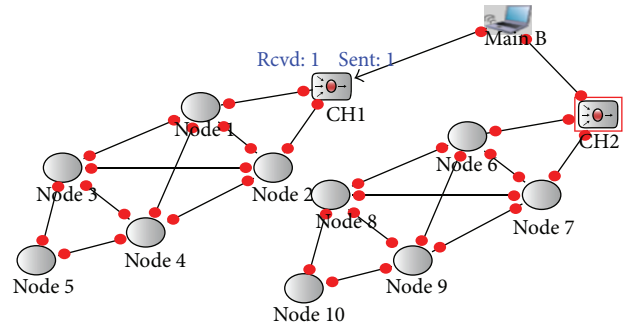

Figure 12: First step in the authentication of L-PKI (first scenario).

Table 5: Energy costs of MICAz, TelosB, and MICA2DOT [16, 19].

Energy cost	MICAz	TelosB	MICA2DOT
Compute for $1T_{clk}$	3.5 nJ (1)	1.2 nJ (1)	
Transmit 1 bit	0.60 μJ (170)	0.72 μJ (600)	59.2 μJ/byte = 7.4 μJ/bit.
Receive 1 bit	0.67 μJ (190)	0.81 μJ (680)	28.6 μJ/byte = 3.5756 μJ/bits.

Table 6: Cost of computation operation.

Energy cost	MICAz	TelosB
AES-128 128-bit encrypt	38 μJ	9 μJ
ECC-160	55 mJ	17 mJ
ECDSA-160 sign	52 mJ	15 mJ
ECDSA-160 verify	63 mJ	19 mJ

cost of any model such as MICAz [22], MICA2DOT [23], and TelosB [30]. MICAz mote is a third-generation module based on the low power 8 bit ATmega128L microcontroller with a clock frequency of 7.37 MHz [19, 22]. It is used to enable low power, wireless sensor network. MICAz runs TinyOS and embed 2.4 GHz IEEE 802.15.4 radio with a claimed data rate of 250 kbps. MICAz offers hardware security (AES-128) [22]. TelosB is a mote module based on the low power 16 bit MSP430 microcontroller with 10 kB RAM and a clock frequency of 4 MHz [19, 24]. TelosB runs TinyOS and embed IEEE 802.15.4 radio with data rate of 250 kbps. Table 5 shows the power which MICAz, TelosB, and MICA2DOT needs for computation and transmission operations [16, 19]. Table 6 illustrates the cost of each computation operation. In Table 6, MICAz running at 7.37 MHz and TelosB at 4 MHz for application data rates, respectively 108 kbps and 75 kbps [19]. Also Table 5 shows the equivalence number of cycles of computation, which is indicated in parenthesis for each operation.

We are going to use the MICAz and TelosB [19] models to calculate the required power. According to [19], the transmission of a single bit of data requires 0.60 μJ and 0.67 μJ for reception. We can calculate the energy cost of *L-PKI* by calculating the energy required for the execution of cryptographic operation and the energy needed for transmitting the required data. We can conclude the energy consumption of L-PKI for each node in Table 7.

Table 7: Total energy cost of L-PKI for each node.

Node	Operation	Required energy	
		MICAz	TelosB
Transmission	Send digital certificate	$192 * 0.6\,\mu J = 115.2\,\mu J$	$192 * 0.72\,\mu J = 138.2\,\mu J$
	Receive neighbor's digital certificate	$192 * 0.67 = 128.6\,\mu J$	$192 * 0.81\,\mu J = 155.5\,\mu J$
	Send signature	$320 * 0.6\,\mu J = 192\,\mu J$	$320 * 0.72 = 230.4\,\mu J$
	Receive neighbor's signature	$320 * 0.67 = 214.4\,\mu J$	$320 * 0.81\,\mu J = 259.2\,\mu J$
	Send or receive session key	$128 * 0.67\,\mu J = 85.76\,\mu J$	$128 * 0.81\,\mu J = 103.68\,\mu J$
	Send OK message	$16 * 0.6\,\mu J = 9.6\,\mu J$	$16 * 0.72\,\mu J = 11.52\,\mu J$
	Receive OK message	$16 * 0.67\,\mu J = 10.72\,\mu J$	$16 * 0.81\,\mu J = 12.96\,\mu J$
	Total cost	**756.28** μJ	**911.46** μJ
Computation	Verification operation of certificate	63 mJ	19 mJ
	Encryption/decryption session key	55 mJ	17 mJ
	Total cost	**118** mJ	**36** mJ
Total		**118.756** mJ	**36.911** mJ

Table 8: Total energy cost of SECMRP for each node.

Energy cost	MICAz	TelosB
Secure route discovery	56.667 mJ	18.934 mJ
Secure data transmission	845.54 μJ	908.28 μJ

7.2. SECMRP Experimentation Results. SECMRP is tested in two different scenarios. The first one was with 10 sensor nodes, two CHs and one BS. The second scenario was with 5 sensor nodes, one CHs and one BS. In both scenarios, nodes are connected with a channel that has a data rate of 250 Kbps and 100 Mbps for CHs and base stations. In the rest of this section, we are going to concentrate on the first scenario.

Energy Cost in SECMRP. We calculated the energy cost of SECMRP by calculating the energy required for the execution of cryptographic operations and the energy needed for transmitting the required data.

SECMRP has three phases secure route discovery, secure data transmission, and route maintenance. We are going to calculate the energy cost of each phase in SECMRP. We can conclude the energy consumption of SECMRP for each node in Table 8.

As we can see from Table 8, secure data transmission phase requires just 845.54 μJ in MICAz and 908.28 μJ in TelosB.

8. Discussion and Analysis

8.1. Protocol Security. The proposed protocols provide several security services, including mutual node authentications, confidentiality, and integrity for sensitive data, while balancing energy cost and performance. The proposed protocols prevent several attacks, such as passive attacks by dividing the sensed data into two parts and encrypting these parts. The protocols utilize multipath to make it difficult for the attacker to capture the whole data. The protocols protect against impersonation or spoofing attacks by invoking mutual authentication using certificates. Modification of protocol messages attacks can be avoided by authentication using nodes' certificates and integrity using MAC. The proposed method clusters the network and uses the certifications to avoided routing table overflow attacks. Replay attacks can be avoided in the proposed method using timestamp and checking the integrity.

Table 9: L-PKI, TinyPK, and μPKI comparison based on security services.

	Authentication	Confidentiality	Nonrepudiation	Scalability
L-PKI	√	√	√	√
TinyPK	√	√		
μPKI		√		√

8.2. Comparison with Related Works. Our proposed L-PKI protocol provides enhancments over other related work, like TinyPK and μPKI. In μPKI, the BS plays the main role in creating the session keys between each pair of nodes. As a result, a lot of traffic is created. In addition, a somewhat weak authentication exists between nodes in the network (as discussed in the Section 2). Table 9 provides a comparison between L-PKI, TinyPK, and μPKI, based on security services.

Our proposed SECMRP protocol enhances several security issues over other route discovery protocols like SEEM and INSENS. SECMRP uses the concept of multipath and clustering to deal with security and efficiency. SECMRP enhances the security issue of SEEM and at the same time uses two disjointed path to send sensed data to the CH. In SEEM, BS works as a server, which floods the query to the network and the node which satisfies the query will send a request to BS for a path, but CH in SECMRP periodically sends two paths to each node. Table 10 provides a comparison between SECMRP, SEEM, and INSENS based on security services.

9. Conclusion and Future Work

Our work improves existing related work by integrating a lightweight PKI with a secure data transmission service via multipath in an efficient manner in terms of resource and power consumption. The proposed protocols enhance the security of WSN by providing mutual authentication between

Table 10: SECMRP, SEEM, and INSENS comparison based on security services.

	Authentication	Confidentiality	Integrity	Balancing	Scalability
SECMRP	√	√	√	√	√
SEEM				√	
INSENS	√	√	√	√	

neighbor nodes and sending the data via multipath. We have provided extensive simulation and evaluation for the proposed protocols and contrasted them with related work. The results are encouraging.

In future work, we plan to enhance the MBFS algorithm to optimize power efficiency as much as possible. We also plan to actually deploy the proposed algorithms on actual sensors in a typical WSN environment.

Acknowledgment

This work is funded by the National Plan for Science and Technology at King Saud University, Project no. 11-INF1500-02.

References

[1] J. Zheng and A. Jamalipour, *Wireless Sensor Networks: A Networking Perspective*, IEEE Press-Wiley, Hoboken, NJ, USA, 2009.

[2] J. Zhang and V. Varadharajan, "Wireless sensor network key management survey and taxonomy," *Journal of Network and Computer Applications*, vol. 33, no. 2, pp. 63–75, 2010.

[3] J. Sen, "A survey on wireless sensor network security," http://arxiv.org/abs/1011.1529.

[4] F. Amin, A. Jahangir, and H. Rasifard, "Analysis of public-key cryptography for wireless sensor networks security," *Proceedings of World Academy of Science, Engineering and Technology*, vol. 31, 2008.

[5] G. Gaubatz, J. P. Kaps, and B. Sunar, "Public key cryptography in sensor networks-revisited," in *Security in Ad-Hoc and Sensor Networks*, pp. 2–18, 2005.

[6] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and RSA on 8-bit CPUs," in *Cryptographic Hardware and Embedded Systems-CHES 2004*, pp. 925–943, 2004.

[7] R. A. Mollin, *An Introduction to Cryptography*, Chapman & Hall, Boca Raton, Fla, USA, 2006.

[8] M. Faisal, J. Al-Muhtadi, and A. Al-Dhelaan, "Towards efficient security services in wireless sensor networks," in *Computer Applications for Bio-technology, Multimedia, and Ubiquitous City Communications in Computer and Information Science*, vol. 353, pp. 114–123, 2012.

[9] B. Kadri, M. Feham, and A. M'hamed, "Lightweight PKI for WSNs uPKI," *International Journal of Network Security*, vol. 10, no. 3, pp. 194–200, 2010.

[10] V. S. Miller, "Use of elliptic curves in cryptography," in *Proceedings of the Advances in Cryptology (CRYPTO '85)*, H. C. Williams, Ed., vol. 218, pp. 417–426, Springer, Berlin, Germany, 1986.

[11] R. Watro, D. Kong, S. F. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," in *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, pp. 59–64, October 2004.

[12] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[13] W. Diffie and M. E. Hellman, "New directions in cryptography," in *Secure Communications and Asymmetric Cryptosystems*, pp. 143–180, Westview, Boulder, Colo, USA, 1982.

[14] N. Nasser and Y. Chen, "SEEM: secure and energy-efficient multipath routing protocol for wireless sensor networks," *Computer Communications*, vol. 30, no. 11-12, pp. 2401–2412, 2007.

[15] J. Deng, R. Han, and S. Mishra, "INSENS: intrusion-tolerant routing for wireless sensor networks," *Computer Communications*, vol. 29, no. 2, pp. 216–230, 2006.

[16] A. S. Wandert, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom '05)*, pp. 324–328, March 2005.

[17] E. Noroozi, J. Kadivar, and S. H. Shafiee, "Energy analysis for wireless sensor networks," in *Proceedings of the 2nd International Conference on Mechanical and Electronics Engineering (ICMEE '10)*, vol. 2, pp. 382–386, August 2010.

[18] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, New York, NY, USA, 2004.

[19] G. De Meulenaer, F. Gosset, F. X. Standaert, and O. Pereira, "On the energy cost of communication and cryptography in wireless sensor networks," in *Proceedings of the 4th IEEE International Conference on Wireless and Mobile Computing, Networking and Communication (WiMob '08)*, pp. 580–585, October 2008.

[20] OMNeT++ Community, http://www.omnetpp.org/.

[21] The AES-CMAC Algorithm, http://tools.ietf.org/html/rfc4493.

[22] MICAz, http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html.

[23] MICA2DOT, http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html.

[24] TELOSB, http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html.