Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2013, Article ID 108968, 17 pages
http://dx.doi.org/10.1155/2013/108968

Research Article

Efficient and Secure Routing Protocol Based on Encryption and Authentication for Wireless Sensor Networks

Jiliang Zhou

Shanghai University of International Business and Economics, Shanghai 201620, China

Correspondence should be addressed to Jiliang Zhou; hnzhoujl@126.com

Received 5 November 2012; Revised 10 March 2013; Accepted 26 March 2013

Academic Editor: Anfeng Liu

One concerned issue in the routing protocol for wireless sensor networks (WSNs) is how to provide with as much security to some special applications as possible. Another is how to make full use of the severely limited resource presented by WSNs. The existing routing protocols in the recent literatures focus either only on addressing security issues while expending much power or only on improving lifetime of network. None of them efficiently combine the above-mentioned two challenges to one integrated solutions. In this paper, we propose efficient and secure routing protocol based on encryption and authentication for WSNs: BEARP, which consists of three phases: neighbor discovery phase, routing discovery phase, and routing maintenance phase. BEARP encrypts all communication packets and authenticates the source nodes and the base station (BS), and it ensures the four security features including routing information confidentiality, authentication, integrity, and freshness. Furthermore, we still design routing path selection system, intrusion detection system, and the multiple-threaded process mechanism for BEARP. Thus, all the secure mechanisms are united together to effectively resist some typical attacks including selective forwarding attack, wormhole attacks, sinkhole attacks, and even a node captured. Our BEARP especially mitigates the loads of sensor nodes by transferring routing related tasks to BS, which not only maintains network wide energy equivalence and prolongs network lifetime but also improves our security mechanism performed uniquely by the secure BS. Simulation results show a favorable increase in performance for BEARP when compared with directed diffusion protocol and secure directed diffusion protocol in the presence of compromised nodes.

1. Introduction

A wireless sensor network (WSN) is a collection of nodes that can form a network without the need of a fixed infrastructure, which operates in an unattended, sometimes hostile, environment. Nodes can be connected arbitrarily, and all nodes take part in discovery and maintenance of routes to other nodes in the network [1]. Thus, one concerned issue when designing wireless sensor network is the routing protocol that requires the researchers to provide as much security to the application as possible [2]. Another important factor makes full use of the severely limited resource presented by WSNs, especially the energy limitation. Current presenters for routing protocols in sensor networks optimize for the limited capabilities of the nodes and the application specific nature of the networks, or they incorporate security into these proposed protocols; however, they have not been designed with security as a goal. When the defender has the liabilities of insecure wireless communication, limited node capabilities, and possible threats, and the adversaries can use powerful laptops with high energy and long-range communication to attack the network, therefore, designing a secure routing protocol for WSNs is crucial and nontrivial [3, 4].

1.1. Background. There are two secure problems to be considered when designing a secure routing protocol. On the one hand, different from the special router between conventional networks connected by wire cable, any node in sensor networks can be a router which can not only route to another node but also receive and send any routing information in a certain scope. On the other hand, one aspect of sensor networks that complicates the design of a secure routing protocol is in-network aggregation and inside attacks. In more conventional networks, a secure routing protocol is typically only required to guarantee message availability. Message confidentiality, authenticity, integrity, and freshness are handled at a higher layer by an end-to-end security mechanism. End-to-end security is possible in more conventional networks because it is absolutely unnecessary for intermediate routers to have access to the content of messages [5, 6]. However, in sensor networks, in-network processing makes end-to-end security mechanisms harder to deploy because intermediate nodes need direct access to the content of the messages. Link layer security mechanisms can help mediate some of the resulting vulnerabilities, but it is not enough for WSNs: we will now require much more from conventional routing protocols.

1.2. Related Works. In this section, we will discuss directed diffusion (DD) protocol and secure DD protocol (S-DD), possible attacks on routing protocol, securing routing protocols, and detecting compromised nodes. DD protocol and secure DD protocol, as our research and comparison representative, are very important, even a milestone, for the research of routing protocol for WSNs, and above all secure routing must exert to aim at some kinds of possible attacks. Moreover, it is necessary for the whole security mechanism to be improved by detecting the compromised nodes in case the routing protocol fails. The following is the current related work for them.

1.2.1. DD Protocol and S-DD Protocol. DD protocol [7] consists of several elements: interests, data messages, gradients, and reinforcements. An interest message is a query or an interrogation which specifies what a user wants. Each interest contains a description of a sensing task that is supported by a sensor network for acquiring data. In general speaking, in DD protocol, the setup and maintenance of extensive routing table are avoided. Instead, it relies on the broadcast propagation of queries, pruned by information content and geographical data. Sensor nodes maintain route caches which contain the source routes of the other nodes that are known. All in all, DD protocol is not resource aware or resource adaptive and especially suffers from many attacks for lack of encryption and authentication in course of packet receiving and transmitting.

Then, Wang et al. [8] present the design of a new secure directed diffusion protocol (S-DD), which provides a secure extension for the directed diffusion protocol. They mainly focus on secure routing and give a simple scheme to securely diffuse data, which uses an efficient one-way chain and do not use asymmetric cryptographic operations in this protocol. However, S-DD cannot work against any active attackers or compromised nodes in the network with the in-network aggregation. Especially, all sensor nodes do not have the ability to authenticate their neighbor nodes, so S-DD is also not robust without the in-network aggregation.

1.2.2. Possible Attacks on Routing Protocol. Two kinds of attacks can target routing protocols for WSNs [9]: passive attacks, where the attacker just eavesdrops on the routing information, and active attacks, where the attacker impersonates other nodes, drops packets, modifies packets, launches denial of service attacks, and so forth.

Most of the current routing protocols assume that all nodes in the network are trustworthy. The control information in the header of the packets carries the routing information, and intermediate nodes are assumed not to change this information. However, a compromised node can easily change the routing field of the packet and redirect the packet to anywhere it wants. The attacker can also redirect the route by changing the route sequence number in some protocols. In that case, the attacker can divert the traffic to itself by advertising a route to a node with the base station (BS) sequence number which is greater than the BS node's route. Redirecting the traffic can also be established by modifying the hop count. Route length is represented as hop count in routing protocol. A compromised node can direct all the traffic to itself by broadcasting the shortest hop count.

These attacks include impersonation, fabrication, and wormhole attacks. Compromised nodes can also create loops by changing the routes in the data packets. This will result in denial of service attacks. In impersonation, the attacker pretends to be another node after learning its address and changes it to its own address. Fabrication is another attack, where the compromised node generates false route messages, such as false error messages. In DD protocol, when links go down and routes break, the node which precedes this broken link broadcasts a "route error message." A compromised node can easily send false error messages for a working route. Another attack is the route cache poisoning attack. Any node can overhear the traffic, and if it finds route information, it adds it to its cache for future use. A compromised node can then broadcast spoofed packets with source route via itself. Then, neighboring nodes hear this and add the route to their cache. Also a compromised node can attack the routing table by overflowing it. It can attempt to initiate route discovery to nonexisting nodes. The worst attack is node captured, in which all information may be exposed and decrypted.

Finally, multihop routing in WSNs causes the packets to be delivered between one or more intermediate nodes. The security of routing information is harder to manage in this case.

1.2.3. Secure Routing Protocols for WSNs. Secure routing protocols for WSNs are difficult to be designed, especially when the nodes of a wireless sensor network have limited resources such as low battery power, CPU processing capacity, and memory. Since most routing protocols currently assume that nodes are trustworthy, security in WSNs mainly deals with authentication of the user nodes and security of the data packets that are being routed. Authentication is one goal, which verifies the identity of a node. A BS, a key, or the use of certificates can be implemented to perform authentications. Certificates can be thought of as a unique identification for every node. In Internet of Things, security mechanisms based on access control and secret communication channels regarding defending against outside attackers have been studied [10].

Zhou and Haas [11] proposed the idea of distributing a BS throughout the network in a threshold fashion. However, Zhou and Haas adopted public key and threshold cryptography, which are very expensive for sensor devices. Therefore, we do not consider this method practical for the time being. All the protocols below assume the preexistence and presharing of secret keys for all honest nodes in the beginning.

Adrian Perrig and Robert Szewczyk [3] present a suite of security protocols optimized for sensor networks: SPINS [5]. SPINS has two secure building blocks: SNEP and μTESLA. SNEP includes data confidentiality, two-party data authentication, and evidence of data freshness. μTESLA provides authenticated broadcast for severely resource-constrained environments. However, their system requires synchronized clocks for all the nodes in the network, and the SPINS is not robust for routing attacker because it is not based on the secure routing protocol.

Secure routing protocol (SRP) proposed by Papadimitratos and Haas guarantees correct route discovery [12]. They assume a security association between the end points in the beginning. The correctness of their protocol was only proven analytically.

Nasser and Chen proposed an efficient routing protocol, which we called SEEM [13], for WSNs. Compared to other proposed routing protocols, SEEM is designed based on utilizing multipath concept and considers energy efficiency and security simultaneously. However, SEEM is not really secure because its packets can be modified by attackers without any encryption and authentication.

Lee and Choi have presented SeRINS [14]: a secure alternate path routing in sensor networks. Their alternate path scheme makes the routing protocol resilient in the presence of compromised nodes that launch selective forwarding attacks. It also detects and isolates the compromised nodes, which try to inject inconsistent routing information, from the network by neighbor report system. In neighbor report system, a node's route advertisement is verified by its surrounding neighbor nodes so that the suspect node is reported to the BS and is excluded from the network. We think the SeRINS has not combined the authentication with encryption, and cooperation of several neighbor nodes can make the reported information good in order to cheat the BS, so the packet of the verified itself is not secure, and this leads the whole protocol not to be secure and trusted.

1.2.4. Detecting Compromised Nodes.

Compromised nodes in WSNs usually promise to forward packets but later drop the data packets and refuse to forward them. Current network protocols do not have a mechanism to detect such nodes. Link layer acknowledgment such as IEEE 802.11 MAC protocol can detect link layer failure. However, it cannot detect a forwarding failure. Some protocol acknowledgments can detect end-to-end communication failure, but it cannot detect which particular node caused the failure in between [15].

Some researchers propose the idea of having neighbor nodes detect each other's behaviors and then report to each other or to a network authority, which detects compromised nodes by observing the reports on several attacks in the network [14, 16, 17]. All nodes have a monitor and reputation records, trust records, and a path manager. All these adapt to changes in networks and find out the misbehaving nodes in the network. However, we think that compromised nodes acting in groups can make these records good for themselves without the authentication mechanism with encryption. Therefore, we believe that a special agent such as BS is necessary for intrusion detection system (IDS) to detect the compromised nodes.

1.3. Contributions.

In this paper, we propose a new routing protocol BEARP: efficient and secure routing protocol based on encryption and authentication for WSNs. In BEARP, we design to encrypt all communication packets, authenticate the source node and the BS, and ensure the four security features including routing information confidentiality, authentication, integrity, and freshness. Moreover, BEARP mitigates the load of sensor nodes by transferring routing-related tasks to the BS which operates routing paths selection and intrusion detection system. In routing paths selection system, the BS periodically selects a newly best path from many paths based on current energy level of nodes along each path. In the process of selecting route, especially, we design the algorithm *multi_shortest_path* to create another child thread, which executes the function *send_rout*e in time when finding a route to the source node. This thread helps decrease the delay for sending routing information. In intrusion detection system, detecting compromised nodes also performs uniquely by the secure BS. Therefore, the two approaches not only maintain network wide energy equivalence and prolong network lifetime but also improve our security mechanism. BEARP can effectively resist to some typical attacks including selective forwarding attack, wormhole attacks, sinkhole attacks, and even a node captured.

Compared to other proposed routing protocols, BEARP not only considers integration between energy efficiency and security simultaneously but regards security as our design goal for the first time. At the same time, the feature making BEARP distinct is that BEARP takes full advantage of the predominance of the BS. As a result, packet delivery ratios and network lifetime for operating BEARP in the WSN are more preferable and work better against some attacks, compared to operating DD protocol. The contributions of our work include the following: (1) we implement the four security features for WSNs including routing information confidentiality, authentication, integrity, and freshness, and BEARP works well under some typical attacks; (2) BEARP has much better packet delivery ratio than DD protocol in the presence of some compromised nodes; (3) the network lifetime is prolonged compared to insecure routing protocols, like DD protocol; (4) BEARP has almost no blocked nodes in WSNs and remarkably surpasses DD protocol.

1.4. Organization of the Paper.

Foregoing contents are our preliminary work before we propose BEARP. The following in this paper is organized as follows. Some used notations and assumptions are introduced in Section 2. In Section 3, we present BEARP routing protocol and related algorithm and give some implementation details. Then, we discuss the security analysis for BEARP in Section 4, followed by performance evaluation in Section 5. Finally, we draw our conclusions in Section 6.

2. Notations and Assumptions

Sensor networks typically consist of one or multiple base stations and hundreds or thousands of inexpensive, small, and hardware-constrained nodes scattered over a wide area. Our sensor network model includes a powerful BS and numerous constrained sensor nodes. BS, which has greater capabilities, can directly transmit data to any node in the network. Resource-constrained sensor node, whose transmission range is limited, can send data along the multihop route to the BS. We consider that a BS is trustworthy, differently from sensor nodes. Moreover, we can extend naturally our scheme for a single BS to multiple BS as presented by Deng et al. [18].

Developing a proper threat model against our routing protocols, we consider two attack sources: outer or insider [19]. Outsider attackers do not have trusted keys. They typically rely on message replay or delay to influence routing protocols. Insider threats occur when a fully trusted node, with appropriate key material, is compromised. We assume the key management system is always secure, since there have been a lot of successful researches for them. The attacks launched from outsiders cannot join in the network because of the assumption, but we consider that the outsider attackers can interfuse in the network to be compromised nodes through any other special means.

Before presenting our BEARP protocol, we introduce some used notations and assumptions about sensor network in Tables 1 and 2, respectively, which are used in the following sections.

3. Routing Protocol Based on Encryption and Authentication (BEARP)

Now we present our BEARP protocol, which consists of three phases: neighbor discovery phase, routing discovery phase, and routing maintenance phase. In the following, each of them will be described in detail [13, 20, 21].

3.1. Neighbor Discovery Phase. Neighbor discovery takes place right after the deployment of all sensor nodes. However, neighbor discovery can be launched at any time by the BS during the lifetime of the sensor network. By doing this, the BS can request to reconstruct this network topology according to the great changes of the topology [13, 20].

To initiate the neighbor discovery, the BS selects broadcast key *BK* to encrypt the packet neighbor discovery (*ND*) and broadcasts the packet confidential *ND* (*CND*) to the whole network. After receiving this packet, each node does as follows (see Table 4):

(1) decrypt $ND = D_{BK}(CND)$ with the broadcast key *BK* of the node;

(2) record the address *prev_hop* from which the current node receives the packet and stores it in the list *neighbor_list* in ascending order of packet received time;

(3) change the address *prev_hop* to the address of itself;

(4) check if the broadcast packet has been received by searching *pkt_seq_num* in the table *rc_pkt_table*. If the packet has already been received once, the node drops this *CND* and does not rebroadcast it. Otherwise, it stores *pkt_seq_num* in table *rc_pkt_table*, encrypts $CND = E_{BK}(ND)$ with the broadcast key *BK,* and rebroadcasts the *CND* to its neighbor.

The fourth step insures that no *CND* packet is broadcasted more than one time for each node, which also applies to other control messages. Thus, the communication overheads for transmitting control packets are reduced to a low level.

Through the process of receiving, decrypting, segmenting, encrypting, and rebroadcasting *CND*, each node knows its real neighbor and stores them for using in the following phases.

The BS waits for a short time to ensure that the *CND* broadcast can be flooded through the network. Then, the BS broadcasts another packet confidential neighbor collection (*CNC*) in order to collect the neighbor information of each node. At the same time, the BS sets the current time T_B and the random number R_B to the packet *NC* in order to authenticate each node in the WSN. After receiving this packet, each node does as follows (see Table 5(a)):

(1) decrypt $NC = D_{BK}(CNC)$ with the broadcast key *BK* of the node and gets the two fields T_B and R_B for creating the reply packet *CNCR*;

(2) check if the address *prev_hop* from which the current node receives the packet has been saved in the list *neighbor_list*. If not then it stores it;

(3) change the address *prev_hop* to the address of itself;

(4) check if the broadcast packet has been received by searching *pkt_seq_num* in the table *rc_pkt_table*. If the packet has already been received once, the node drops this *CNC* and does not rebroadcast it. Otherwise, it stores *pkt_seq_num* in table *rc_pkt_table*, keeps the other fields in the packet, encrypts $CNC = E_{BK}(NC)$ with the broadcast key *BK,* and rebroadcasts the *CNC* to its neighbor.

When sensor node receives the *CNC* packet, it replies a confidential neighbor collection reply (*CNCR*) packet to the BS by flooding. In *NCR*, we add the session key field *SK*, time field T_B, and random number field R_B, and the source address is set to itself, and the destination address is set to the BS. The *CNCR* packet contains the following information:

(1) the address of the node,

(2) the list that has all addresses of its neighbors,

(3) the session key between the node and the BS,

(4) the authentication information.

Table 1: Basic notations.

BK	The initial key used to create the session key between BS and nodes and encrypt the routing message at beginning.
SK	Session key used for data encryption and authentication between BS and source.
T_B, T_S	Denote the current time of the BS and the current time of the source node, respectively.
R_B, R_S	Denote the random number of the BS selected and the random number of the source node selected, respectively.
$E_k(x)$	Encryption of message x with key k.
$D_k(x)$	Decryption of cipher message x with the key k.
$x\|y$	Concatenation of message x and y.
prev_hop	Denote the previous node address from which the current node receives the packet
next_hop	Denote the next node address to which the current node sends the packet.
neighbors_list	Neighbors address list.
pkt_seq_num	The sequent number of a packet.
rc_pkt_table	Received packet table that stores the sequent number of packets.
route_list	The routing list field in a packet.
route_table	The routing table in a node.
pkt_type	Packet type including *CND*, *CNC*, *CNCR*, *CDE,* and *CDER*.
source_add	The source node address.
data_length	The length of data packet.
itself_add	The current node address.
$A \xrightarrow{M} B$	Node A sends message M to node B.

Table 2: Assumptions.

A-1:	The links between these sensor nodes are always bidirectional. The communication patterns in WSNs fall into three categories: node to BS, BS to node, and BS to all nodes.
A-2:	The BS has sufficient battery power to surpass the lifetime of all sensor nodes and sufficient memory to store cryptographic keys, and it is very secure and cannot be compromised under any conditions.
A-3:	Each node in WSNs has unique identifier stored in BS, and it can forward a message towards the BS, recognize packets addressed to it, and handle message broadcasts.
A-4:	WSNs may be deployed in unauthentic locations, and basic wireless communication is not secure. Individual sensors are untrustworthy; any adversary can eavesdrop on traffic, inject wrong routing messages, and replay old routing messages.
A-5:	Each node can get a master secret key which it shares with the BS before its deployment. The secret key is used as authentication key by the BS.
A-6:	The BS can update all secret keys between any nodes after a certain period of time, and the key management system is always secure.

Table 3: Neighborhood matrix.

	BS	1	2	3	4	5	6
BS	0	∞	∞	0	0	0	0
1	1000	0	1000	1000	1000	0	0
2	1000	1000	0	1000	1000	0	0
3	0	1000	1000	0	1000	1000	1000
4	0	1000	1000	1000	0	1000	1000
5	0	0	0	1000	1000	0	1000
6	0	0	0	1000	1000	1000	0

Each node receiving this packet does as follows (see Table 5(b)):

(1) decrypt $NCR = D_{BK}(CNCR)$ with the broadcast key BK of the node;

(2) check if the address *prev_hop* from which the current node receives the packet has been saved in the list *neighbor_list*. If not then it stores it;

(3) change the address *prev_hop* to the address of itself;

(4) check if the broadcast packet has been received by searching *pkt_seq_num* in the table *rc_pkt_table*. If the packet has already been received once, the node drops this *CNCR* and does not rebroadcast it. Otherwise, it stores *pkt_seq_num* in table *rc_pkt_table*, keeps the other fields in the packet, encrypts $CNCR = E_{BK}(NCR)$ with the broadcast key BK, and rebroadcasts the *CNCR* to its neighbor.

When the BS receives the packet *CNCR*, at first, it must authenticate communication time cost by comparing the time field T_B with the current time and authenticate the freshness of the packet by comparing random number field R_B with the foregone random number R_B. If the authentication fails, the BS will drop the packet. Finally, after receiving neighbor information of all nodes, the BS has a vision of the topology of the whole networks.

To select a path that has the maximum available energy on each node, we introduce the concept weight. The weight of an edge in the corresponding graph of the network is

Table 4: Confidential neighbor discovery packet broadcasts.

BS (sender)		Neighbor of BS (receiver)
$ND = [pkt\_type \| BS \| prev\_hop \| pkt\_seq\_num]$; $CND = E_{BK}(ND)$.	$\xrightarrow{\text{CND}}$	$ND = D_{BK}(CND); prev\_hop \rightarrow neighbor\_list$; $itself\_address \rightarrow prev\_hop$; $pkt\_seq\_num \rightarrow rc\_pkt\_table$; $ND = [pkt\_type \| BS \| prev\_hop \| pkt\_seq\_num]$; $CND = E_{BK}(ND)$.
N_i (sender)		N_{i+1} (receiver, neighbor of node N_i)
$ND = D_{BK}(CND); prev\_hop \rightarrow neighbor\_list$; $itself\_address \rightarrow prev\_hop$; $pkt\_seq\_num \rightarrow rc\_pkt\_table$; $ND = [pkt\_type \| BS \| prev\_hop \| pkt\_seq\_num]$; $CND = E_{BK}(ND)$.	$\xrightarrow{\text{CND}}$	$ND = D_{BK}(CND); prev\_hop \rightarrow neighbor\_list$; $itself\_address \rightarrow prev\_hop$; $pkt\_seq\_num \rightarrow rc\_pkt\_table$; $ND = [pkt\_type \| BS \| prev\_hop \| pkt\_seq\_num]$; $CND = E_{BK}(ND)$.

Table 5: Confidential neighbor collection and confidential neighbor collection reply packet broadcasts.

(a) Confidential neighbor collection packet broadcasts

BS (sender)		N_i (receiver)
$NC = [pkt\_type \|$ BS $\| prev\_hop \| pkt\_seq\_num \| T_B \| R_B]$; $CNC = E_{BK}(NC)$.	$\xrightarrow{\text{CNC}}$	$NC = D_{BK}(CNC)$; get T_B, R_B; $prev\_hop \rightarrow neighbor\_list$; $itself\_address \rightarrow prev\_hop$; $pkt\_seq\_num \rightarrow rc\_pkt\_table$; $NC = [pkt\_type \| BS \| prev\_hop \| pkt\_seq\_num \| T_B \| R_B]$; $CNC = E_{BK}(NC)$.

(b) Confidential neighbor collection reply packet broadcasts

Source node (sender)		N_i (receiver, neighbor of BS)
$NC = D_{BK}(CNC)$; $NCR = [pkt\_type \|$ source_add $\| neighbor\_list \| prev\_hop \| pkt\_seq\_num \| SK \| T_B \| R_B - 1]$; $CNCR = E_{BK}(NCR)$.	$\xrightarrow{\text{CNCR}}$	$NCR = D_{BK}(CNCR)$; $prev\_hop \rightarrow neighbor\_list$; $itself\_address \rightarrow prev\_hop$; $pkt\_seq\_num \rightarrow rc\_pkt\_table$; $NCR = [source\_add \| neighbor\_list \| prev\_hop \| pkt\_seq\_num \| SK \| T_B \| R_B - 1]$; $CNCR = E_{BK}(NCR)$.
N_i (sender)		BS (receiver)
$NCR = D_{BK}(CNCR)\ (prev\_hop) \rightarrow neighbor\_list$; $itself\_address \rightarrow prev\_hop$; $pkt\_seq\_num \rightarrow rc\_pkt\_table$ $NCR = [source\_add \| neighbor\_list \| prev\_hop \| pkt\_seq\_num \| SK \| T_B \| R_B - 1]$; $CNCR = E_{BK}(NCR)$.	$\xrightarrow{\text{CNCR}}$	$NCR = D_{BK}(CNCR)$; $pkt\_seq\_num \rightarrow rc\_pkt\_table$; get $source\_add$, $neighbor\_list$, SK, $R_B - 1$; authenticate T_B, $R_B - 1$.

the available energy on the head node. The BS then constructs a directed graph marked weight with neighbor information. The weight decreases as the head node sends and receives packets. Figure 1 shows the subgraph derived from the network topology. In Figure 1, the weights of edges starting from BS are infinite, which means that the BS has much more energy than other sensor nodes.

The calculation of the weight is based on the formula

$$\text{Weight} = \frac{\text{total power of each node}}{\text{power for transmitting or receiving one packet}}. \tag{1}$$

We assume the total energy of each node initially is 10000 units and the total energy for sending one packet is about 10 units; then

$$\text{weight} = \frac{10000}{10} = 1000 \text{ units}. \tag{2}$$

We use neighborhood matrix to represent the neighborhood relations between nodes. Table 3 shows the weighted matrix corresponding to the graph in Figure 1. Each row except the first row contains the neighbor information of a specific node; for example, the second row shows neighbor information of the BS. Each column except the first column represents a node. If the value for some space is not zero, it means that the nodes corresponding to the row and the

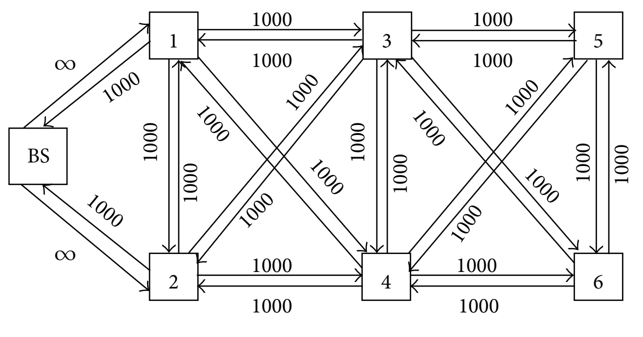


Figure 1: The subgraph derived from network topology.

column are neighbors. The value of each space is the weight of the edge from the node corresponding to the row to the node corresponding to the column. As we defined, weights of edges from the BS are infinite. For example, from Table 3 we know that *node 3* has five neighbors: *node 1, node 2, node 4, node 5,* and *node 6.*

3.2. Routing Discovery Phase. The BS starts its task, routing discovery, beginning at phase two. The task is divided into three subtasks: data enquiry; routing path selection system (RPSS); sending routing information.

3.2.1. Data Enquiry. By now, BEARP supports only data transmission requested by the BS; that is, the BS broadcasts enquiry for data with specific features. Sensor nodes have satisfied the enquiry response with enquiry reply. Data transmission follows these steps.

(1) The BS broadcasts an enquiry packet confidential data enquiry (*CDE*).

(2) Sensor nodes have satisfied the enquiry response with a reply packet confidential data enquiry reply (*CDER*).

(3) Sensor nodes that do not satisfy the enquiry rebroadcast *CDE*.

(4) The BS calculates a shortest path to the desired node in the weighted graph. The shortest path is a path from the source to the BS of which the total energy consumed on each node for sending one packet is the least, that is, usually the path with minimum hops.

Each node receiving *CDE* packet does as follows:

(1) check if it satisfies the enquiry itself;

(2) if not, the node rebroadcasts the *CDE* and saves the *pkt_seq_num* to avoid repeating broadcasting the *CDE* more than once;

(3) if it does, the node returns a *CDER* packet that contains the length of data sent soon to the BS by setting the *next_hop* to the first node in the *neighbor_list*. Because the *neighbor_list* is in the ascending order of the receiving time of *CND* and *CNC*, the first node in the *neighbor_list*, sometimes even the second and the third and so on, must be one-hop close to the BS than the node itself. If the node is the neighbor of the BS, the BS must be the first node in its *neighbor_list*.

In packet *CDER*, we add the length of data field *data_length*, time field T_S, and random number field R_S, and the source address is set to itself, and the destination address is set to the BS. The packet *CDER* contains following information:

(1) the address of the source node, the previous hop, and the next hop,

(2) the length of data,

(3) the authentication information: time field T_S, random number field R_S.

Source node selects a broadcast key *BK* to encrypt the packet *DER*, selects the session key *SK* to encrypt $T_S \parallel R_S$, and broadcasts a confidential *DER* (*CDER*) packet to the whole network. After receiving this packet, each intermediate node does as follows (see Table 6):

(1) decrypt $DER = D_{BK}(CDER)$ with the broadcast key *BK* of the node;

(2) change the address *prev_hop* to the address of itself;

(3) get the address *prev_hop* in the first record of *neighbor_list* to set the *next_hop*;

(4) check if the broadcast packet has been received by searching *pkt_seq_num* in the table *rc_pkt_table*. If the packet has already been received once, the node drops this *CDER* and does not rebroadcast it. Otherwise, it stores *pkt_seq_num* in table *rc_pkt_table*, encrypts $CDER = E_{BK}(DER)$ with the broadcast key *BK*, encrypts $T_S \parallel R_S$ with the session key *SK*, and rebroadcasts the packet *CDER* to its neighbor.

When the BS receives the *CDER* packet, it decrypts $DER = D_{BK}(E_{BK}(CDER))$ with the broadcast key *BK* and $T_S \parallel R_S = D_{SK}(E_{SK}(T_S \parallel R_S))$ with the session key *SK*. It gets the *data_length* for the following calculations of the shortest path, through which we can get the energy for sending the data from the source node and random number field R_B and time T_B for authentication.

3.2.2. Routing Path Selection System (RPSS). After the BS receives the packet *CDER*, in order to tell the source node a best routing path to BS, it starts to calculate the shortest path to the node. However, it is important for BS how to calculate the shortest path in WSNs. We design routing path selection system (RPSS) to solve the problem.

As mentioned above, the shortest path has the minimal sum of energy consumed for transmitting one packet, that is, usually the path with minimum hops. Thus it saves the energy from the view of the whole network. When there are more than two shortest paths, we use the maximal available power as the second criteria; that is, we select the path that has the maximal available energy on each sensor node.

To get the desired shortest path, we modify the breadth first search (BFS) algorithm [5] to get the relatively shortest

Table 6: Confidential data enquiry reply packet forwards.

Source node (sender)		Intermediate nodes (receiver)
$DER = [pkt\_type \| source\_add \| data\_length \| prev\_hop \| next\_hop \| pkt\_seq\_num]$; $CDER = E_{BK}(DER) \| E_{SK}(T_S \| R_S)$	$\xrightarrow{\text{CDER}}$	$DER = D_{BK}(E_{BK}(CDER))$; $itself\_address \rightarrow prev\_hop$; First record of $neighbor\_list \rightarrow next\_hop$; $pkt\_seq\_num \rightarrow rc\_pkt\_table$; $DER = [pkt\_type \| source\_add \| data\_length \| prev\_hop \| next\_hop \| pkt\_seq\_num]$; $CDER = E_{BK}(DER)) \| E_{SK}(T_S \| R_S)$
		The base station
	$\xrightarrow{\text{CDER}}$	$DER = D_{BK}(E_{BK}(CDER))$ $T_S \| R_S = D_{SK}(E_{SK}(T_S \| R_S))$; Get data_length.

path from the BS to source node, as is shown in Algorithm 1. The BFS always finds the shortest path from the source to the destination, if there is one. Our modified version of BFS algorithm does not necessarily select the absolute shortest path because we also need to consider the left energy, that is, the weight corresponding to each edge, of each node into consideration. That is, if one node on the shortest path has energy left less than required level, we discard this shortest one and continue searching the second shortest path until success.

We assume that the BS wants to get data from source node N. We first define three levels of energy limitation. Each level is the half of the upper level. The main modification to the breadth-first search algorithm is that whenever it finds a shortest path to source node N, it checks if the weight of each edge on the path is greater than the predefined level. If so it returns this path as the shortest path. Otherwise, it continues the calculations until it finds the second shortest path. If the shortest path under current energy limitation cannot be found, it means that each path found has at least one node whose energy level is less than current energy limitation. Consequently, we degrade the energy limitation to the lower level and search again. If not any path is found from the first level to the third level, it means that source node N is unreachable [20, 21].

In a word, BS maintains an energy limitation array for all nodes, and the updating of energy limitation for each node is independent. This feature ensures the best use of each node in the sensor network. In the RPSS, BS can determine whether it has the routing path to the source node or not and how many routes it may be selected. If there are routes to the source, the BS will select the shortest route and send to it in time.

3.2.3. Sending Routing Information. In the algorithm *multi_shortest_path*, we introduce into multiple-threaded process mechanism. As is to know that a thread is a lightweight process which exists within a program and executed to perform a special task in operating system. A process that has only one thread is referred to as a single-threaded process, while a process with multiple threads is referred to as a multiple-threaded process [22]. In our design, a thread is placeholder information associated with a single use of a program that can handle multiple concurrent users, and several threads of execution may be associated with a single process. In runtime environment designed by us, some threads exist in a common memory space and can share both data and code of a program, and they can increase the speed of any application.

We then present the process of executing the related function of RPSS. When any standalone application is running, it first executes the method *main* running in a one thread, called the main thread. The main thread creates another child thread which executes the function *send_route_to_source_node*, and the function schedules another function *multi_shortest_path*, which urgently creates another child thread which executes the function *send* when finding a route to the source node. Algorithm 2 shows code segments for sending route to source node algorithm. The method *main* execution can be finished, but the program will keep running until all threads have completed its execution. As is shown in Algorithms 1 and 2.

The multiple-threaded process mechanism mentioned above evidently decreases the delay with the multiple-threaded process because it can satisfy with multiple users and concurrent requests. If multiple users are using the function *multi_shortest_path*, the threads are created and maintained for each of them. Our design not only increases the speed of selecting a path to the source but also always saves memory space.

Once the BS has got the routing path to the source, the route is set to the field *route_list* of the packet RR. The BS then selects a broadcast key BK to encrypt the packet route reply (RB) including the route to the source, selects the session key SK to encrypt T_S and $R_S - 1$, and sends the confidential RR (CRR) packet to the second address of the route. Each intermediate node forwards this packet according to the corresponding of the route.

When the source node receives the CRR, at first, it must authenticate communication time cost by comparing the time field T_B with the current time and also authenticate the freshness of the packet by comparing random number field R_B with the foregone random number R_B. If the authentication fails, the source node can conclude that the sender of the packet is not real or the route is not credible and drops the packet. Otherwise, the source node stores *route_list* in table *route_table* and *pkt_seq_num* in table *rc_pkt_table*.

Multiple shortest path algorithm

```
vector<nsaddr_t> BEARP::multi_shortest_path(nsaddr_t
source_node, int packet_energy){
    bool visited[NODES_AMOUNT];
    nsaddr_t father[NODES_AMOUNT];
    nsaddr_t tmp, neighbor_node;
    vector<nsaddr_t> queue, route [PATHS_AMOUNT];
    bool found;
    int pointer, path_amount;

    found = false;
    path_amount = 1;
    nsaddr_t father_node;
    queue . reserve(NODES_AMOUNT);
    route . reserve(NODES_AMOUNT);

    while (!found){
        pointer = 0;
        for (int i = 0; i <NODES_AMOUNT; i++){
                visited[i] = false;
                father[i] = −1;
        }
        for (int i = 0; i <queue.size(); i++)
            queue · pop_out();

        visited[BASE_STATION] = true;
        queue . reserve(NODES_AMOUNT);
        queue . push_in((nsaddr_t)BASE_STATION);

        for (int i = 0; i <NODES_AMOUNT; i++){
            if (left_energy[i]− packet_energy >=
                current_energy_limit[source_node]){
                for (int j = 0; j<NODES_AMOUNT; j++){
                    if (weight_matrix[i][j] > 0 && weight
                            _matrix[i][j] < MAX_INT){
                      if (left_energy[j]− packet_energy >=
                            current_energy_limit[source_node])
                            all_neighbor_list[i] . push_in(j);
                      }
                }
            }
        }


        tmp = queue[pointer];
        visited[tmp] = true;
        while(queue · size() > pointer){
            for(int i = 0; i<all_neighbor_list[tmp] . size(); i++){
                neighbor_node = all_neighbor_list[tmp][i ];

                if (visited[neighbor_node])
                continue;
                father [neighbor_node] = tmp;

                if (neighbor_node == source_node){
                    father_node = neighbor_node;
                    while(father_node!= BASE_STATION){
                        route . push_in(father_node);
                        father_node = father[fa];
                    }
                    found = true;

                    if(path_amount = 1){
                      path_amount++;
                      threadbegin /* create a thread to send route*/
                      send(route[0], source_node);
```

Algorithm 1: Continued.

```
                                threadend
                            }
                        }
                        visited[neighbor_node] = true;
                        queue . push_in(neighbor_node);
                    }
                    pointer++;
                    tmp = queue[pointer];
                }
                if (found && path_amount = =PATH_MAX)
                    break;
                update_energy_limit(source_node,
                        current_energy_limit[source_node]);
            }
            return route;
        }
```

Algorithm 1: Code segments for multiple shortest path algorithm.

```
Send_route_to_source_node(nsaddr_t source_node, int packet_energy){
    /* start multi_shortest_path algorithm */
    vector<nsaddr_t> BEARP::multi_shortest_path(nsaddr_t source_node, int packet_energy);
    bool exchange;
    int high, low, path_amount;
    multi_shortest_path (nsaddr_t source_node, int packet_energy);
    count(int timer); exchange = true;
    high = path_amount; low = 1;
    while(timer >= TIME_MAX && listening(ACK) = false){
        if(high >= low){
            if(exchange){
                exchange = false;
                path_amount −−;
                send(route[high −−], source_node);
            }
}
            else{
                exchange = true;
                path_amount −−;
                send(route[low + + ], source_node);
            }
        }
        else {
            multi_shortest_path(nsaddr_t source_node,
              int packet_energy);
            high = path_amount; low = 1; exchange = true;
        }
    }
    Return(1);
}
```

Algorithm 2: Code segments for sending route to source node.

Table 7 shows the process of forwarding the confidential route reply packet in the WSN.

At the same time, the *ACK* mechanism can also help the source node find a correct route to the BS. On receiving the *CRR* packet, the source node knows which path it can use to communicate with the BS. As a result, using the path transferred with the *CRR* packet, it returns an *ACK* packet to the BS to confirm the receipt of the *CRR*. The *ACK* packet also contains the number of data packets going to be sent, which to some extent guarantees that the receiver can detect the loss

TABLE 7: Confidential route reply packet forwards.

BS (sender)		Intermediate nodes (receiver)
$DER = D_{BK}(E_{BK}(CDER))$;		$RR = D_{BK}(E_{BK}(RR))$;
$T_S \Vert R_S = D_{SK}(E_{SK}(T_S \Vert R_S))$;	$\xrightarrow{\text{CRR}}$	*The corresponding of route_list* $\rightarrow$ *next_hop*;
$RR =$ *[pkt_type‖source_add‖route_list‖pkt_seq_num]*;		*pkt_seq_num* $\rightarrow$ *rc_pkt_table*;
$CRR = E_{BK}(RR) \Vert E_{SK}(T_S \Vert R_S - 1)$.		$CRR = E_{BK}(RR) \Vert E_{SK}(T_S \Vert R_S - 1)$.
		Source node
		$RR = D_{BK}(E_{BK}(CRR))$;
	$\xrightarrow{\text{CRR}}$	*route_list* $\rightarrow$ *route_table*;
		pkt_seq_num $\rightarrow$ *rc_pkt_table*;
		authenticate $T_S \Vert R_S - 1$: $T_S \Vert R_S - 1 = D_{SK}(E_{SK}(T_S \Vert R_S - 1))$.

of data due to communication problems, nodes failure, or misbehavior of compromised nodes. After sending the *ACK* packet, source node is ready to start transmitting the real data. If the BS does not receive the *ACK* packet within a predefined time, it deems the selected route as invalid and runs the function *multi_shortest_path* once more to find another path. If it receives the *ACK* packet, then it knows that this route is available and waits for data from the source node.

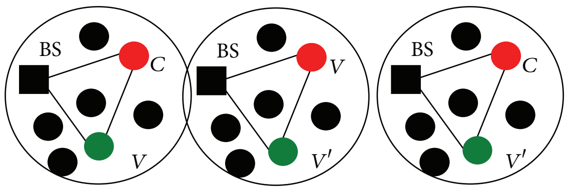


FIGURE 2: Network of including malicious nodes.

3.3. Routing Maintenance and IDS. In route maintenance, it is still the BS that works as the server to operate intrusion detection system (IDS) and to release control information. The purpose of the phase route maintenance is to overcome this potential risk by IDS and to prolong the network lifetime as much as possible [16–18, 23].

The BS must verify all nodes entering the network at the beginning. This is one of our assumptions. In our proposed solution, the BS detects any compromised node which possibly exists in the network by impersonating regular users. The BS detects the compromised node by sending each node arbitrary route requests one by one. Figure 2 shows a network of including compromised nodes. When a BS wants to test whether a node (let us say node C) is forwarding other nodes' data packets inside the network or not, the BS will first pick a validated destination node V that is close to node C. Then, the BS will send a *REQ* to node C for node V. Once node C agrees to participate in the route and a route is established between the BS, node C, and node V, the BS will send data packets to node V using this route. Then, by sending information and asking for the received packets to node V encrypted with its private shared key between the BS and node V, the BS will check whether node V has received the packets or not. Node V will send back an acknowledgment to BS whether it has received any data packets from node C or not. If it has not, the BS will test whether the node C and V are forwarding other nodes' data packets inside the network or not, so the BS will continue to pick another validated destination node V' and node V. Thus, the BS will check whether node V' has received the packets or not by sending information and asking for the received packets to the validated destination node V' encrypted with its private shared key between the BS and node V'. If it has not, the BS will mark node C or V as compromised nodes and will update the network key immediately. Algorithm 3 shows code segments for detecting compromised node algorithm in intrusion detection system.

All nodes in the network except for the compromised nodes will receive the new network shared key. From that the compromised nodes will not be able to encrypt or decrypt any packet information [24]. The BS will know that the nodes are compromised and take them out of the network. At the same time, the BS will process the routing tree including the compromised nodes.

4. Security Analysis of BEARP

In this section, we will analyze the security properties of BEARP required by sensor networks and present how BEARP defends some typical attacks in the WSN.

4.1. Routing Message Confidentiality. A sensor network should not leak sensor readings, especially control packet, to neighboring networks. We have assumed that the key management system is secure, which is the underlying security for our BEARP, so the secret keys are confidential. The standard approach for keeping sensitive routing message secret is to encrypt them with a secret key that only intended receivers possess, hence our BEARP can distinctly achieve routing message confidentiality. Given the observed communication patterns, we set up secure channels between nodes and base stations and later bootstrap other secure channels as necessary.

4.2. Identity Authentication and Routing Message Authentication. Since an adversary may exert to personate or imitate a compromised node, identity authentication and routing message authentication are important for many applications in sensor networks [25]. The receiver needs to ensure that

```
Detect_compromised_node(compromised node $C$, validate node $V$ )
/* Use the validate node $V$ to confirm whether the node $C$ is compromised node or not.* /
            { if (Base Station received the acknowledge packet of the validate node)
                  return (true);
              else {
          select another validate node $V'$;
              if (detect_compromised_node(compromised
                  node $V$, validate node $V'$))
                validate node $C$ is compromised node;
                else
              {
              detect_compromised_node(compromised
                node $C$, validate node $V'$);
              validate node $V$ is compromised node;
              }
          }
    }
```

Algorithm 3: Code segments for detecting compromised node algorithm.

the routing message used in any decision-making process originates from a trusted source. Informally, routing message authentication allows a receiver to verify that the routing message was really sent by the claimed sender. In the two-party communication case of the BEARP, routing message authentication can be achieved through a purely symmetric mechanism: the sender and the receiver share a session secret key to compute the four particular parameters (T_B, R_B, T_S, and R_S) of all communicated routing message because they are correlative with the routing message. When a routing message with four correct particular parameters arrives, the receiver knows that it must have been sent by the sender. If the four particular parameters have some mistakes, the receiver concludes the sender or intermediate node may be adversary.

4.3. Routing Message Integrity. In communication, routing message integrity ensures the receiver that the received routing message is not altered in transmission by an adversary [5]. In BEARP, we achieve routing message integrity through routing message authentication for the four particular parameters (T_B, R_B, T_S, and R_S), which is not a stronger property. It is very difficult that an adversary only alters routing information but does not alter the four particular parameters because the routing message is confidential as a whole. At the same time, the packet sequence number *pkt_seq_num* can also help authenticate routing message integrity.

4.4. Routing Message Freshness. Routing message freshness means that the routing message is recent, and it ensures that no adversary replayed old messages. Sensor networks send measurements over time, so it is not enough to guarantee confidentiality and authentication [5]. In BEARP, to ensure each routing message is fresh, we design a real-time T_B or T_S field of the routing packet, which provides to conclude the freshness of the packet through computing and comparing the two particular parameters (T_B, T_S), the receiving time, allowing delay time.

4.5. Defending Some Typical Attacks. The most direct attack against a routing protocol is to target the routing information exchanged between nodes. By spoofing, altering, or replaying routing information, adversaries may be able to create routing loops, attract or repel network traffic, extend or shorten source routes, generate false error messages, partition the network, increase end-to-end latency, and so forth. [9]. Apparently, routing information in BEARP which holds the above four security properties can defend adversaries to spoof, alter, or replay them.

Wormholes are hard to detect because they use a private, out-of-band channel invisible to the underlying sensor network. Sinkholes are difficult to defend against in protocols that use advertised information such as remaining energy or an estimate of end-to-end reliability to construct a routing topology because this information is hard to verify [26].

However, resistant to the two attacks is the most important of all secure targets of our designing the routing protocols [25]. Adversary cannot encrypt and decrypt the routing information with the secret key, and it cannot pretend to be another node to impersonate and fabricate any other information. Furthermore, all routing paths are selected uniquely by the BS which is very secure and cannot be compromised under any condition in our assumption. Therefore, protocols that construct a topology initiated by a BS are most susceptible to wormhole and sinkhole attacks, but BEARP can easily defend them.

In a selective forwarding attack, compromised nodes may intend to include themselves on the actual path of the data flow and refuse to forward certain messages and simply drop them, ensure that they are not propagated any further. However, once again the mechanism that BEARP selects routing paths prevents sensor nodes from selecting or joining routing path. All routing paths are selected uniquely by the BS, which defends adversaries to join in the WSN. A more subtle form of this attack is when an adversary selectively forwards packets. An adversary interested in suppressing or modifying packets originating from a select few nodes can reliably forward the remaining traffic and limit suspicion of its wrongdoing. Routing message confidentiality can prevent adversary to open any routing packets.

When an adversary captures a sensor node in WSNs and knows all its secret keys, BEARP also has two methods for secure process: one is routing paths selected uniquely by the secure BS, which reject a sensor node captured to imitate BS; another is IDS of detecting compromised nodes, which can take the sensor node captured out of the WSN.

5. Performance Evaluations and Analyses

The goal of our experiments is to evaluate and analyze the performance of our BEARP. To simplify the simulation, we generated random nodes and defined some of them as compromised nodes. In BEARP, these compromised nodes are kicked out of the network as soon as they discovered; however, in DD protocol and S-DD protocol, these nodes are not detected. In our simulations, no compromised nodes will be allowed to reenter the network before being certified by the BS, and therefore they will not be able to route packets again. In the following sections, We measured the packet delivery ratios, network lifetime, and nodes blocked by compromised attacks during data forwarding, which are very important for efficient and secure routing protocol for WSNs [3], and we then show the simulation results for different scenarios.

5.1. Simulation Metrics. To evaluate the performance of our secure routing mechanism in the presence of some compromised nodes which impact network performance, we have simulated BEARP on a network simulator, ns-2 [27]. In our simulations, we consider to generate a variety of sensor fields of different sizes. Some sensor nodes, ranging from 100 to 1200, are randomly deployed in $200 \times 200\,\text{m}^2$ target area, and the network size is changeable according to different measure for network performance. Regarding the left-bottom corner of the target area as (0, 0), we positioned a BS at a fixed point (100, 100), almost in the center of the WSN. Each sensor node has a constant transmission range of 20 m. All sensor nodes are stable, and no node is moving, and every round each node sends 20 packets to the BS. We changed the scenario files each time for testing the BEARP protocol, DD, and S-DD protocol for different numbers of nodes, compromised nodes.

5.2. Packet Delivery Ratios. In this scenario, we increased the compromised nodes into the WSN for every test case. The simulation time was 90 s in test. In Figure 3, we show the packet delivery ratio when there are some compromised nodes amounts from 10 to 100 present in the WSN. As we can see from the figure, the BEARP has better packet delivery ratio than the DD protocol and S-DD protocol all the time. This is due to the fact that since compromised nodes are left out of the network because of encryption and authentication in BEARP, all data packets may not be sent to them. Therefore, the packet delivery ratios of the BEARP hold rather higher than those of the DD protocol and S-DD protocol.

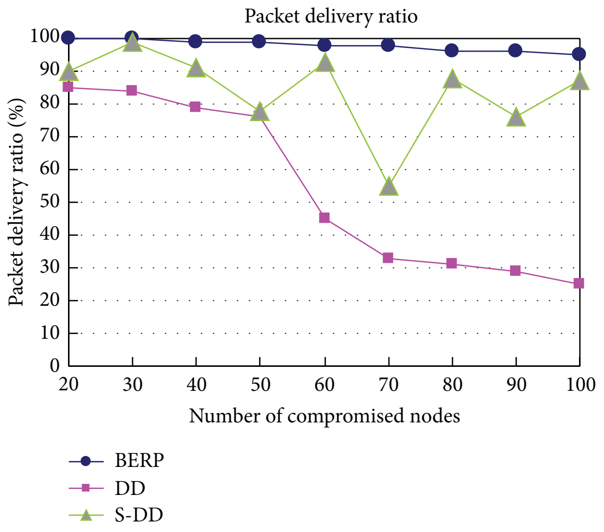
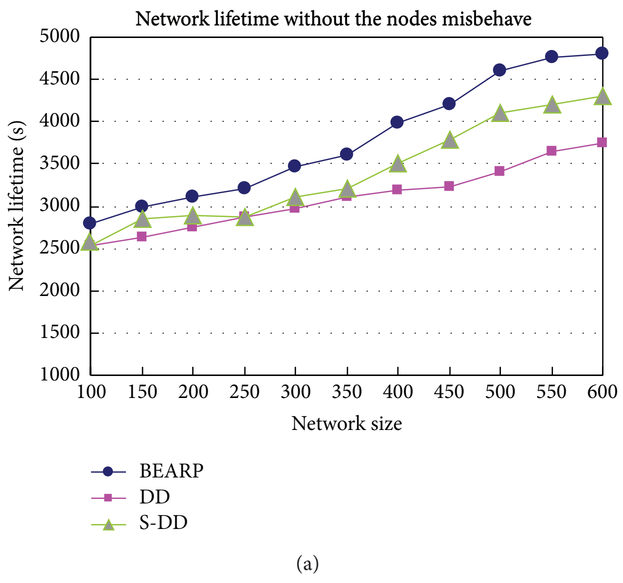


Figure 3: Packet delivery ratio (%) for 600 nodes.

At the same time, as the number of compromised nodes increases, the packet delivery ratio for both protocols goes down because the compromised nodes are dropping the packets. Especially, the packet delivery ratio for DD protocol descends sharply when the compromised nodes are more than 50. In S-DD protocol, without authentication mechanism between neighbors, the compromised node may transmit or not when the packets send to a compromised node. Thus, the packet delivery ratio for S-DD protocol is unstable. We think that since certain compromised nodes are chosen randomly, there is a chance that compromised nodes may occupy crucial positions for data transferring. In BEARP, there are not many nodes left in the WSN since the compromised nodes are being left out due to the mechanism of encryption and authentication. Therefore, it is taking a certain time to establish connections and for the packets to be delivered, and the packet delivery ratio for BEARP decreases slightly.

5.3. Network Lifetime. The most significant performance increase achieved in BEARP is the network lifetime. In Figure 4(a), we can see that BEARP increases the network lifetime over 15% and 8%, respectively, compared to DD protocol and S-DD protocol. Though the rule for reinforcing a particular path differs, it is always the fact that DD protocol and S-DD protocol use the same path for all communications between the same source and BS. The direct consequence is that nodes on this particular path may deplete energy very soon, while BEARP uses several shortest paths and maintains an energy limitation array for all nodes to avoid each node to exhaust energy quickly. Figure 4(b) is the simulation results for network lifetime when 10% of nodes misbehave. From this figure we can see that network lifetime of DD protocol suffers a significant decrease, and S-DD protocol's lifetime is increased but unstable while that of BEARP decreases slightly and be stable. When compromised nodes destroy the path for forwarding, both DD protocols and S-DD protocol



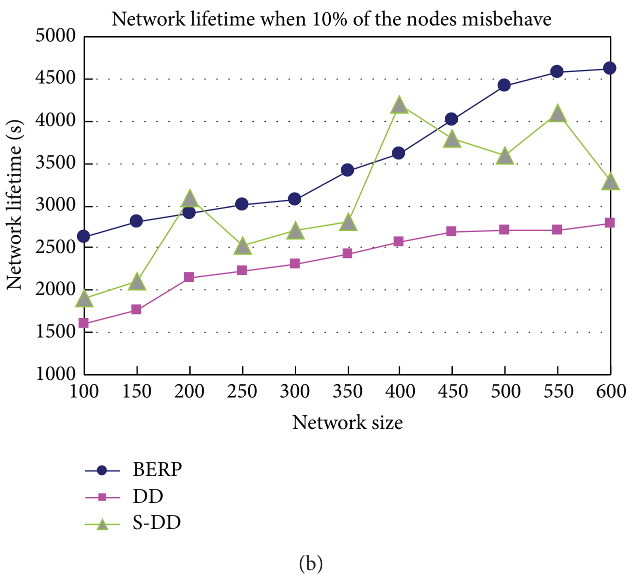


Figure 4: Compared average network lifetime between BEARP, DD, and S-DD protocol. (a) without the nodes misbehaving. (b) when 10% of the nodes misbehave.

have to select a new path, and the communication load spreads among a small number of available paths. Instability of lifetime for S-DD protocol is that it cannot detect and reject compromised nodes. Moreover, the lifetime of BEARP is 37% longer than DD protocol because BEARP can reject compromised node, resist their attacks, and distribute the load more evenly among several secure paths according to the algorithm *multi_shortest_path*.

5.4. Nodes Blocked by Compromised Attacks. We randomly distributed compromised nodes over the square area. In the simulations, we considered two types of compromised nodes: one drops all the relaying packets (type I), and the other drops all the relaying packets and also advertises inconsistent routing information (type II). In addition, we simulated two different density networks: one is 600 sensor nodes network, and the other is 1200 sensor nodes network.

We performed a set of experiments to measure the number of sensor nodes blocked by a set of compromised nodes in each round, increasing the number of compromised nodes in the network. In the presence of type I compromised nodes, we, respectively, measured the number of blocked nodes running on the BEARP, on the DD protocol, and on the S-DD protocol in both 600 and 1200 sensor nodes networks. We also measured the number of blocked nodes using the same scheme in the presence of type II compromised nodes.

Each simulation experiment was conducted using 10 different network topologies, and each result was averaged over 10 runs of different network topologies.

Simulation experiment results are shown in Figure 5. In the presence of type I compromised nodes which drop all the relaying packets, the effect of DD protocol, S-DD protocol, and our BEARP on a ratio of blocked nodes is shown in Figures 5(a) and 5(b). S-DD protocol is not stable to be blocked by type I compromised nodes. In 600 sensor nodes network, using DD protocol incurs blocked nodes from about 5% to 44%, while BEARP has almost no blocked nodes for compromised node to be entered to WSNs due to the secure authentication mechanism, as shown in Figure 5(a). In Figure 5(b), in 1200 sensor nodes network, using DD protocol has less blocked nodes than in 600 sensor nodes network. This is because, the number of sensor nodes scattered in the network is doubled, which makes the network denser. Also, each sensor node has more neighbor nodes so that it has more next-hop nodes. Thus, this increases the chances of bypassing the compromised nodes which drop relaying packets.

In the presence of type II compromised nodes which both drop all the relaying packets and advertise inconsistent routing information, the effect of secure authentic system on a ratio of blocked nodes is shown in Figures 5(c) and 5(d). Without secure authentic system, the influence of type II compromised nodes over the network is more devastating than that of type I nodes, since type II compromised nodes even attract the network traffic and drop them. Using secure authentic system, however, we see that more than 99% of sensor nodes are not blocked, as shown in Figures 5(c) and 5(d). Since, in the experiments, almost every type II nodes were excluded by secure encryption and authentication system from the network; legitimate nodes did not forward packets to the compromised nodes identified. Thus, with several type II nodes, almost all of them are excluded from the network so that more than 99% of sensor nodes are not blocked.

Out of control is the cause of network blocked. In WSN with compromised nodes, DD protocol cannot control the relaying for any packets, while S-DD protocol cannot control the relaying of neighbor's packets due to no secure authentic system. On the one hand, our secure authentic system in BEARP can protect the sensor nodes from compromising

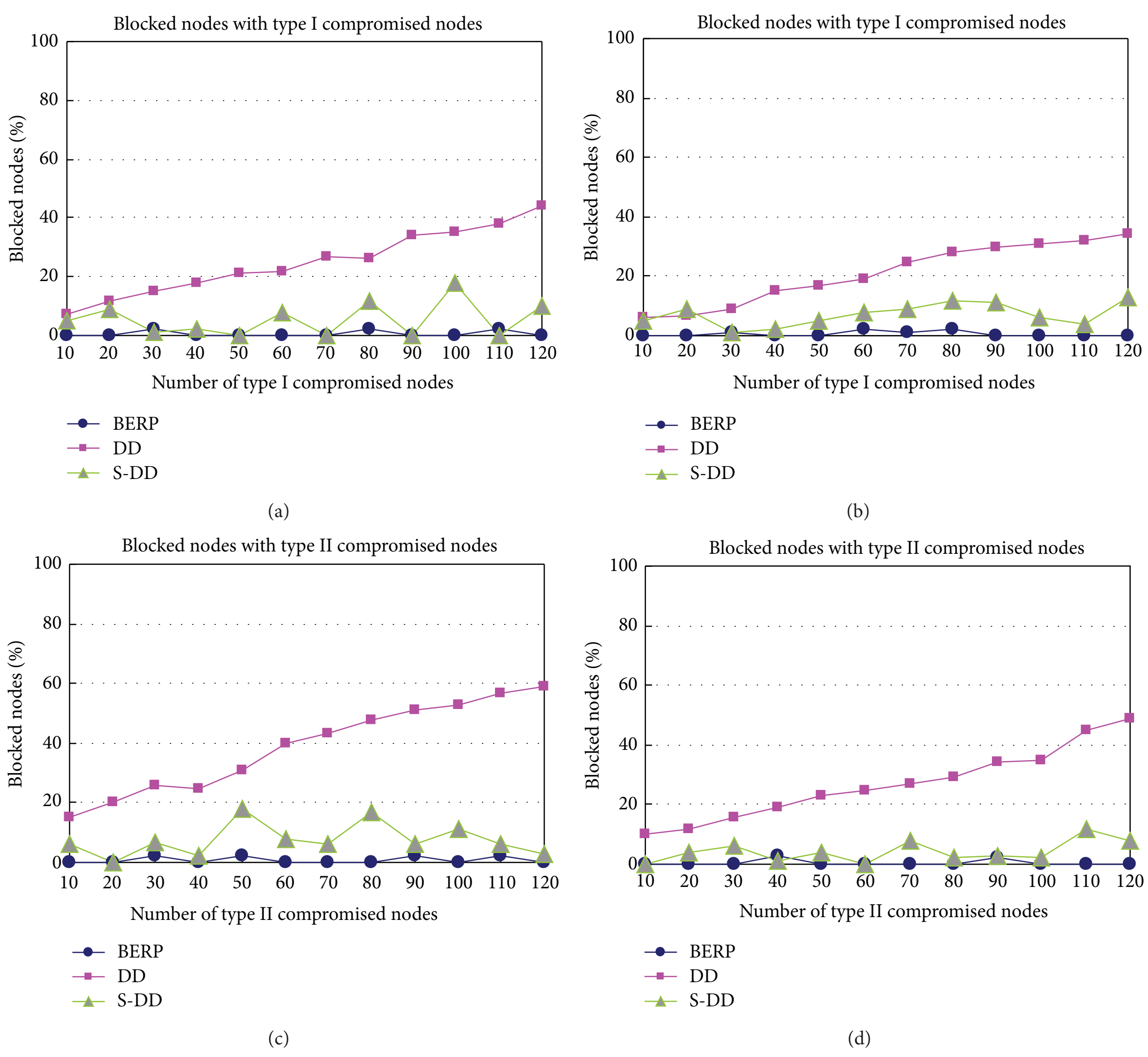


FIGURE 5: Performance evaluation for nodes blocked by compromised attacks (average over 10 runs). (a), respectively, executes BEARP, DD, and S-DD protocol in 600 sensor nodes network, in the presence of type I compromised nodes; (b), respectively, executes BEARP, DD, and S-DD protocol in 1200 sensor nodes network, in the presence of type I compromised nodes; (c), respectively, executes BEARP, DD, and S-DD protocol in 600 sensor nodes network, in the presence of type II compromised nodes; (d), respectively, executes BEARP, DD, and S-DD protocol in 1200 sensor nodes network, in the presence of type II compromised nodes.

in WSNs. On the other hand, even if the sensor nodes suffer insurmountable attacks of compromised nodes, the IDS including the algorithm *detect_compromised_node* has an ability to detect type I and type II compromised node, and makes them become no more a member of the network so that they cannot influence other legitimate nodes any more. However, seen in Figures 5(b) and 5(d), as the network gets denser and each node's degree becomes higher, our BEARP makes the network more resilient in the presence of type I or type II compromised nodes.

6. Conclusions

Nowadays, most of the wireless sensor network routing protocols are implemented with no security in mind. Incorporating security into these protocols can only solve some simple security problems, so we focus on security mechanisms for the WSN and design a security routing protocol as a goal, which is performed throughout the network. Simultaneity, using the power of network nodes for security is a necessary evil. Consequently, we propose the efficient and secure routing protocol called BEARP.

We presented the BEARP absolutely different from the well-known DD protocol and the other routing protocol incorporated security. BEARP can successfully not only achieve routing message confidentiality, authentication, integrity, and freshness but detect the compromised nodes in a network by IDS. We implemented an encryption and authentication mechanism to encrypt the data packets between any two nodes and authenticate BS and source node in the network. Moreover, BEARP has two methods for secure process: one is routing paths selected uniquely by the secure BS; another is our important IDS for detecting compromised node. All the secure mechanisms are united together to make our routing protocol BEARP effectively resilient in the presence of compromised nodes that launch selective forwarding attacks, wormhole attacks, sinkhole attacks, and even a node captured.

At the same time, we also make full use of the severely limited resource presented by WSNs, especially the energy limitation. Our BEARP mitigates the loads of sensor nodes by transferring routing-related tasks such as RPSS and IDS to the BS, which not only efficiently maintains network wide energy equivalence and prolongs network lifetime but also successfully improves our security mechanism. Especially, in algorithm *multi_shortest_path* of the RPSS, we design the multiple-threaded process mechanism, which not only increases the speed of selecting a path to the source but also always saves memory space and the contents of the register when RPSS is interrupted and restored. Furthermore, RPSS maintains an energy limitation array for all nodes, and the updating of energy limitation for each node is independent. This feature ensures the best use of each node's energy in the sensor network.

Simulation results show a favorable increase in the performance evaluation for BEARP when compared to DD protocol in the presence of compromised nodes. Our protocol surpasses the DD protocol and S-DD protocol in terms of the packet delivery ratios, network lifetime, and nodes blocked by compromised attacks during data forwarding.

However, we only considered the efficient and secure routing protocols of BS actively launch. In future work, we will focus the research on security mechanisms for other different WSNs, also for particular misbehaviors of some compromised nodes such as denial-of-service attacks and jamming attacks.

Acknowledgments

The authors would like to thank anonymous referees for their useful comments. This work is supported by the Special Research Funds of Educational Ministry of China under Grant no. 104086.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[2] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005.

[3] A. Perrig and R. Szewczyk, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 9, pp. 534–548, 2002.

[4] A. Quintero, D. Y. Li, and H. Castro, "A location routing protocol based on smart antennas for ad hoc networks," *Journal of Network and Computer Applications*, vol. 30, no. 2, pp. 614–636, 2007.

[5] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[6] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," in *Proceedings of the ACM Workshop on Wireless Security*, pp. 21–30, September 2002.

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 56–67, Boston, Mass, USA, August 2000.

[8] X. Wang, L. Yang, and K. Chen, "SDD: secure directed diffusion protocol for sensor networks," in *Proceedings of the 1st European Workshop (ESAS '04)*, C. Castelluccia et al., Ed., Lecture Notes in Computer Science, pp. 205–214, August 2004.

[9] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[10] R. Roman, C. Alcaraz, J. Lopez, and N. Sklavos, "Key management systems for sensor networks in the context of the internet of things," *Computers & Electrical Engineering*, vol. 37, no. 2, pp. 147–159, 2011.

[11] L. Zhou and Z. J. Haas, "Securing wireless sensor networks," *IEEE Network Magazine*, vol. 6, pp. 30–37, 1999.

[12] P. Papadimitratos and Z. J. Haas, "Secure routing for mobile wireless sensor networks," in *Proceedings of the SCS Communication Networks and Distributed Systems (CNDS '02)*, pp. 27–31, 2002.

[13] 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.

[14] S. B. Lee and Y. H. Choi, "A secure alternate path routing in sensor networks," *Computer Communications*, vol. 30, no. 1, pp. 153–165, 2006.

[15] B. Wu, J. Wu, E. B. Fernandez, M. Ilyas, and S. Magliveras, "Secure and efficient key management in mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 30, no. 3, pp. 937–954, 2007.

[16] Y. C. Hu, A. Perrig, and D. B. Johnson, "Rushing attacks and defense in wireless Ad Hoc network routing protocols," in *Proceedings of the ACM Workshop on Wireless Security*, pp. 30–40, New York, NY, USA, September 2003.

[17] Y. C. Hu, D. B. Johnson, and A. Perrig, "SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 1, pp. 175–192, 2003.

[18] 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.

[19] T. Ito, H. Ohta, N. Matsuda, and T. Yoneda, "A key predistribution scheme for secure sensor networks using probability density function of node deployment," in *Proceedings of*

the ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '05), pp. 69–75, November 2005.

[20] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 255–265, August 2000.

[21] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensornetworks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[22] G. J. Narlikar, "Scheduling threads for low space requirement and good locality," *Theory of Computing Systems*, vol. 35, no. 2, pp. 151–187, 2002.

[23] S. Zhong, J. Chen, and Y. R. Yang, "Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks," in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies*, pp. 1987–1997, April 2003.

[24] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 52–61, October 2003.

[25] H. Yang, F. Ye, Y. Yuan, S. Lu, and W. Arbough, "Toward resilient security in wireless sensor networks," in *Proceedings of the 6th ACM International Symposium on Mobile (MobiHoc '05)*, pp. 34–45, 2005.

[26] M. Abomhara, O. Zakaria, O. Khalifa, A. Zaidan, and B. Zaidan, "Enhancing selective encryption for H. 264/AVC using advanced encryption standard," *International Journal of Computer Theory and Engineering*, vol. 2, no. 2, pp. 223–229, 2010.

[27] G. Chen, J. W. Branch, M. Pflug, L. Zhu, and B. K. Szymanski, "SENSE: a wireless sensor network simulator," in *Advances in Pervasive Computing and Networking*, B. Szymanski and B. Yener, Eds., Springer, New York, NY, USA, 2005.