A Secure Energy-Efficient Routing Protocol for WSN*

Al-Sakib Khan Pathan and Choong Seon Hong

Department of Computer Engineering, Kyung Hee University
1 Seocheon, Giheung, Yongin, Gyeonggi, 449-701, South Korea
spathan@networking.khu.ac.kr, cshong@khu.ac.kr

Abstract. The intent of this paper is to propose an energy-efficient routing protocol with data transmission security for wireless sensor networks. We create an energy and distance aware sink-rooted tree in the network which is used for secure data transmissions from the source sensors to the base station. We mainly focus on ensuring authenticity and confidentiality of the sensor reports by adopting one-way hash chain and pre-loaded shared secret keys. To achieve data freshness, there is an optional key refreshment mechanism in our protocol. Along with the detailed description of our protocol, we present an analysis and performance evaluation of our proposal.

1 Introduction

Wireless sensor networks (WSNs) could be very useful for providing support for some specific purposes like target tracking, surveillance, environmental monitoring etc [1], [2]. Today's sensors can monitor temperature, pressure, humidity, soil makeup, vehicular movement, noise levels, lighting conditions, the presence or absence of certain kinds of objects or substances, mechanical stress levels on attached objects, and other properties. As such types of networks are composed of resource-constrained tiny sensors, many research works have tried to focus on efficient use of the available resources of sensors. Energy is in fact, one of the most critical factors that play a great role to define the duration of an active WSN. So, any protocol should ensure a good structure of the network based on the energy levels of the participating nodes or a competent way of utilizing the energies of the sensors in the network. Energy-efficiency is also very necessary to maximize the lifetime of the network. Security on the other hand, is another critical issue especially for ensuring the legitimacy of transmitted reports from the sensors to the base station (BS) [3], [4]. The task of securing a WSN is however, complicated by the fact that the sensors are mass-produced anonymous devices with a severely limited energy budget and initially with no knowledge of their locations in the deployment environment (in general cases).

In this paper, we deal with the challenge of energy-efficiency and secure routing in WSN in a highly dense deployment scenario. We propose a protocol which aims at minimizing the wasteful energy consumption by energy-efficient structuring of the network and then secure the data transmissions from the sensors to the base station

* This work was supported by the MIC and ITRC. Dr. C.S. Hong is the corresponding author.

using one-way hash chain and shared secret keys. The rest of the paper is organized as follows: Section 2 states the related works, section 3 presents our assumptions and preliminaries, Section 4 describes our protocol in detail, performance evaluation and analysis are presented in section 5, and section 6 concludes the paper.

2 Related Works

[5] proposed an energy-efficient security protocol for wireless sensor network by using symmetric key cryptography and NOVSF code-hopping technique. The basic idea is to implement two algorithms in the sensor nodes and in the base station which the sensor nodes and the base station would follow at the time of transmitting data within the network. To ensure better security they introduced NOVSF technique which basically scrambles the data blocks using a multiplexer in the system while transmitting data from the sensor nodes. [5] is claimed to be secure and energy efficient considering the fact that, it increases the level of security during data transmission using the NOVSF technique without utilizing any additional power. However, this scrambling technique increases the complexity of tasks for the base station. To address the issue of energy efficient data aggregation with secure data transmission, ESPDA protocol [6] is proposed. Ye et al. [7] proposed a statistical en-route filtering (SEF) scheme to detect and drop false reports during the forwarding process. In their scheme, a report is forwarded only if it contains the message authentication codes (MACs) generated by multiple nodes, by using keys from different partitions in a global key pool. Zhu et at. [8] proposed the interleaved hop-by-hop authentication scheme that detects false reports through interleaved authentication. Their scheme guarantees that the base station can detect a false report when no more than *t* nodes are compromised, where *t* is a security threshold. Motivated by [8], Lee and Cho [9] proposed an enhanced interleaved authentication scheme called the key inheritance-based filtering. In their scheme, the keys of each node used in the message authentication consist of its own key and the keys inherited from its upstream nodes. Every authenticated report contains the combination of the message authentication codes generated by using the keys of the consecutive nodes in a path from the base station to a terminal node. Other than these, [10], [11], [12], [22] focused only on energy efficiency and [3], [4], [13] dealt with the security measures for routing in WSN.

3 Network Assumptions and Preliminaries

We consider a WSN with dense deployment of sensors, where initially all nodes and the BS have same transmission ranges and all of their clocks are synchronized. The BS has unlimited resources. The sensors have the resources like the MICA2 motes [14]. Once they are deployed, they remain relatively static in their respective positions. Each node transmits within its transmission range isotropically (in all directions) so that each message sent is a local broadcast. The link between any pair of nodes in the network is bidirectional, that is, if a node n_i gets a node n_j within its transmission range (i.e. one hop), n_j also gets n_i as its one-hop neighbor.

The BS cannot be compromised in any way and no node could be compromised by any adversary while creating the tree structure in the network (i.e., section 4.1). Each sensor has a shared secret key with the BS which is pre-loaded into its memory.

An accurate model for energy consumption per bit at physical layer is given by $E = E_{elec}^{trans} + \beta d^{\alpha} + E_{elec}^{recv}$, where, E_{elec}^{trans} is the distance-independent amount of energy consumed by the transmitter electronics and digital processing, E_{elec}^{recv} is the energy utilized by receiver electronics, while βd^{α} accounts for the radiated power necessary to transmit over a distance *d* between source and destination.

Like [15], [16] we assume that $E_{elec}^{trans} = E_{elec}^{recv} = E_{elec}$. So, overall energy consumption between source and destination within 1-hop is calculated by $E = 2.E_{elec} + \beta d^{\alpha}$. Broadly speaking, hierarchical routing protocols use control packets for topology construction phase. For a particular node *i*, control packet transmission cost is,

$$C_i^{ctrl}(r) = \left[L_{ctrl} \times \beta r^{\alpha} + (nbr_i(r) + 1) \times L_{ctrl} \times E_{elec}\right]\frac{1}{T}$$

where, α: the path loss exponent ($2 < \alpha < 5$), β: a constant [joule/bit.m^2], r: transmission range, L_{ctrl} : length of control packet (bits), nbr_i : number of neighbors of *i* for range r, L_{elec} : energy needed by the transceiver circuitry to transmit/receive a packet and T : the time period between two consecutive restructuring of network.

For a particular path *p*, data communication cost from source *i* to the BS is,

$$C_i^{data}(p) = \left[\sum_{i=1,j=2}^{N} (nfrd_p(d_i)+1) \times L_{data} \times \beta d_{i,j}^{\alpha} + (nbr_p(d_i)+1) \times L_{data}\right] \times E_{elec}$$

Here, N : total number of nodes in the network, $i, j \in 0,1,2,....,N$: node index, *p*: the path associated for data transmission from *i* to sink, d_i : transmission range set by *i*, $d_{i,j}$: distance between the *i* and *j*, $nfrd_p(d_i)$: number of forwarding nodes for a path *p* and range *d*, $nbr_p(d_i)$: the number of neighboring nodes for a path *p* and range *d*, L_{data} : length of data packets (bits), and finally, α and β are same as previous equation. Total communication cost for sending a data packet from source *i* is:

$$C_i^{total}(p) = \sum_{i=1}^{N} \left[C_i^{ctrl}(r)\right] + C_i^{data}(p)$$

The observations from the above equation are, (a) Wasteful (due to idle listening, overhearing etc.) energy consumption (WEC) increases as the number of redundant forwarder increases, (b) WEC increases as the number of idle nodes increases, (c) Energy consumption increases exponentially as the distance increases, and (d) Frequency of control packet transmission is proportional to the energy consumption. To reduce energy consumption following things could be done, (a) Reducing the number of forwarding nodes (keeping the level of connectivity and reliability of network intact), (b) Putting certain portion of the nodes in sleep mode, (c) Employing adaptive transmission range according to the distance from the forwarder to save energy, and, (d) Fixing network restructuring frequency to ensure balanced energy consumption.

We consider three states of the nodes in our protocol during its operation: (a) *Non-forwarding* - the nodes keep their radio transceivers 'Off' but the sensing circuitry 'On', (b) *Forwarding* - both the transceiver and sensing circuits remain 'On' in this state, (c) *Active* – during the tree construction and OHC initialization phase (section 4.1), all nodes remain in the active state. In active state, both the sensing and radio circuitries of the sensors remain 'On'. Basically, there is no major difference between these two states. We use two terms to differentiate two phases in our protocol.

Determining Active State Time. Let, v be a node and $N_1(v)$ be the number of one-hop neighbors of v for a particular transmission range r (r is same for all nodes). Let, T_{rtt} be the round trip time for data propagation between the longest distant pair within one hop neighbors. Then, the active state time for v is given by the equation, $T_{active} = T_{rtt} \times N_1(v)$. In our protocol, within the time T_{active}, a node could be able to determine whether it should participate in the tree as a forwarding node or not.

To ensure security for data transmissions from sensors to BS, we use pre-stored shared secret keys and one-way hash chain. A one-way hash chain [17] is a sequence of numbers generated by one-way function F that has the property that for a given x, it is easy to compute $y = F(x)$. However, given F and y, it is computationally infeasible to determine x, such that $x = F^{-1}(y)$. An one-way hash chain (OHC) is a sequence of numbers $K_n, K_{n-1}, \ldots, K_0$, such that, $\forall i : 0 \leq i < n$, $K_i = F(K_{i+1})$. To generate an OHC, first a random number K_r is selected as the seed, and then F is applied successively on K_r to generate other numbers in the sequence.

4 Our Proposed Protocol

4.1 Tree Construction and OHC Initialization Phase

We create a sink rooted tree (SRT) in the network. During the tree construction, all nodes keep their radio transceivers 'On' to verify whether it should remain active as a forwarder or not. A timer parameter is defined to ensure each node's active participation in this process for a specific period of time. Each node is prioritized for transmission according to its residual energy and distance from the sink.

According to our assumption, all the sensors and the BS have pre-loaded shared secret keys which could be used to provide confidentiality of the reports. However, to provide authenticity of the transmitted data, all the intermediate nodes between any source node and the base station must be initialized with the basic one-way hash chain number. Let us suppose the initial OHC number is $I_{OHC} = HS_0$.

To initiate the network structuring and OHC number initialization phase, the base station B generates a control packet containing HS_0, a MAC (Message Authentication Code) for the control packet using the key K_i, where K_i is the number in the key chain number corresponding to time slot t_i and some other parameters. The format of the control packet is: *bcm*: $B|sid|ren|dist|fid|HS_0|MAC_{K_i}(B|sid|ren|dist|fid|HS_0)$, where, *sid* indicates the sender's id, *ren* is the remaining energy of the sender, *dist* is the calculated cumulative distance to reach the BS using forwarder(s), and *fid* is the id of the upstream node (i.e. immediate parent or immediate forwarding node) selected by the current node for forwarding data towards the sink. The sink node initiates *bcm* with sender id *B*, and the values of *sid*, *ren*, *dist* and *fid* as -1, as according to our assumption, the base station has unlimited energy compared to the energies of the sensors in the network, and in this case, no forwarding node is needed to reach itself.

When the BS transmits *bcm*, at first its one-hop neighbors get the message. Receiving the message, each node in the one-hop neighborhood of the BS first calculates its distance from the base station based on the received signal strength, stores the value of HS_0 and sets *B* as its forwarder node (the ultimate destination is the BS). Now, each of these nodes transmits the message again within its own one-hop neighborhood (i.e., local broadcast). In this case, the *sid* is set to its own id, *ren* is its own residual energy and the MAC part is kept the same as the message, *bcm*.

To ensure prioritization of the transmission of control messages, each node waits for a threshold time before each further transmission. Waiting time of a node before further transmission is defined by the following equation, $T_{wait} = \{D_s/E_r\} \times R$, where, D_s is the cumulative distance between the sink and the node, E_r is the residual energy of the node, and *R* is a constant that is needed to normalize the value of T_{wait} . As with the course of time, the sensors lose their energy levels and the value of the ratio of distance and residual energy increases; we need to normalize this value. In our case, *R* is the ratio of node's initial energy and transmission range.

Each node receiving the control messages from one or more upstream neighboring nodes, first calculates the distance of each sender based on the received signal strength, then calculates the cumulative distances up to the BS via different possible forwarders, stores the id and residual energy information of each sender and stores HS_0 from the message sent by the first sender. To choose its forwarder node, it compares the values of the distance and energy ratios (D_s/E_r) of the neighboring upstream nodes and chooses the node with the least value of the ratio as its forwarder node.

It then senses the channel and if the channel is idle, it waits for T_{wait} time and then re-transmits the message containing its own status information and with its chosen forwarder id as its *fid*. As the selected upstream node could also get the message (because of bidirectional link), it sets itself as a forwarding node for this transmitting node. This process continues and eventually a tree-structure is created in the network where, each node has a forwarder node on the way to reach to the base station and possibly one or more downstream nodes that can send data to it destined to the base station. Here, the value of T_{wait} depends mainly on the values of E_r and D_s.

To authenticate HS_0, *B* releases the key K_i in time slot t_{i+d}. On receiving this key, a node can verify the integrity and source authentication of HS_0. Thus, along each path the initial OHC number is initialized. It is to be noted that, *bcm* won't bring any attack against the network even if the nodes on the other side of the network don't receive K_i at t_{i+d}. Since, the messages that are MACed by K_i are supposed to be sent out at time slot t, an adversary cannot launch any attacks with K_i when it gets K_i at t_{i+d}. Within the time T_{active}, a node which does not get any message from any of its neighbors that, it should be a forwarding node, sets itself as a non-forwarding node.

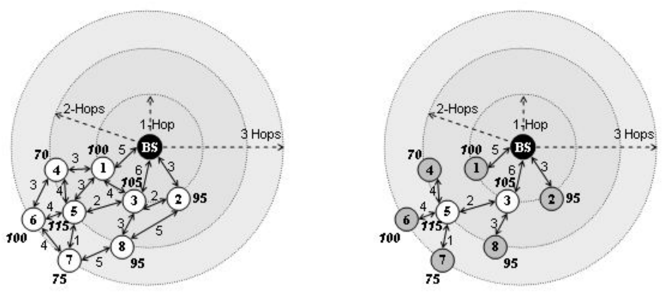

Fig. 1. (left) Before execution of the first phase (right) After execution of the first phase. The gray nodes are in non-forwarding status while the other two nodes are in forwarding status. In these figures, we have shown the N-hop (N = 1, 2, 3 …) neighbors of the BS on the circumference of the same circle regardless of their actual calculated distances from the BS.

Let us illustrate the first phase of our protocol with an example. Figure 1 shows a portion of a network where there are 8 nodes. The black filled circle is the base station and the white circles are the sensors. The bold italic numbers beside each node indicate the residual energies of the nodes and the number in between any pair of nodes indicates the distance between those two nodes. For simplicity here we assume that *R* is 1. The base station initiates the first phase. The 1-hop neighbors of the base station in Figure 1, nodes 1, 2 and 3 get the message first. All of these nodes calculate their T_{wait} and T_{active} values. Here, the T_{wait} values for the nodes 1, 2, and 3 are 0.05, 0.057 and 0.03 respectively. In this dense deployment scenario, let us suppose that the node 2 and 3 are within the transmission ranges of each other, but as node 2 has lower T_{wait} value, it transmits the message before node 3 does. Here, when node 2 transmits the message, node 3 and 8 get the message (as its one hop neighbors), store the id, E_r and D_s of 2 for future computations. In case of node 1 and node 3, node 1 transmits before node 3 can do the transmission. The local broadcast of node 1 is heard by node 3, 4 and 5. Eventually, these nodes store the required information from this message. It should be noted here that, as node 1, 2 and 3 get the control message directly from the base station, they set the base station as their *fid*s.

After node 1 and node 2 have done the transmissions, let us take a sample case. Say, node 3 and 5 are neighbors of each other. Now, node 3 has T_{wait} value 0.057 and node 5 has T_{wait} value 0.069. So, node 3 gets the chance of further transmission before node 5. Now node 8 is a neighbor of node 3. So, node 3's transmission is heard by node 5 and 8. As a 1-hop neighbor of the base station, node 3 does not need to select a forwarder, rather base station is its forwarder. When node 4's turn comes, it chooses node 5 as a forwarder node for itself. Node 5 knows this as it is a 1-hop neighbor of node 4. This process continues, and the whole network is structured as a sink rooted tree where there are several paths from the base station to the leaf nodes. Along the path, the initial OHC number is also initialized. Figure 1(right) shows the resultant structure of the network after the first phase is executed.

4.2 Network Operation and Secure Data Transmission Phase

After the tree is constructed within the network, all nodes are either in forwarding or non-forwarding states. All nodes try to sense any change of parameters within their vicinities and upon detecting any event, the non-forwarding nodes turn their radios on and transmit data towards the base station via their selected forwarding nodes.

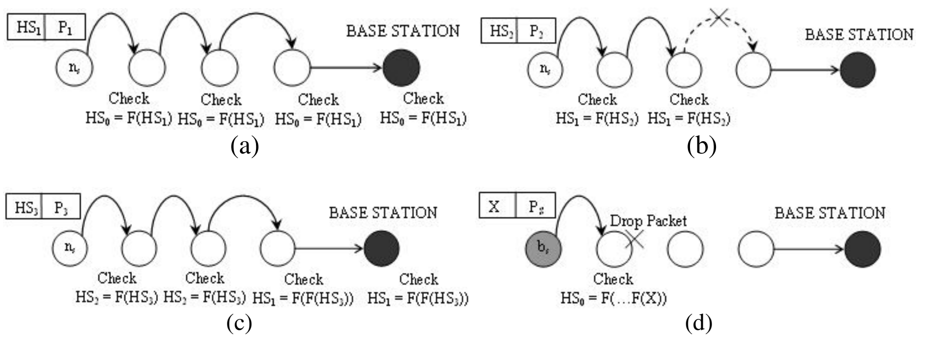

Fig. 2. (a) Authenticated packet delivery to the BS (b) Packet could not reach the BS (c) But it cannot affect the OHC verification technique (d) A bogus packet is dropped

To send the data securely to BS, each source node n_s maintains a unique one-way hash chain, HS: $<HS_n , HS_{n-1}, \ldots , HS_1, HS_0>$. When a source n_s sends a report to the sink using the path created in the sink-rooted tree (for example, $n_s \rightarrow \ldots \rightarrow n_{m-1} \rightarrow n_m \rightarrow B$), it encrypts the packet with its shared secret key with the BS, includes its own id and an OHC sequence number from HS in the packet. It attaches HS_1 for the first packet, HS_2 for the second packet, and so on. To validate an OHC number, each intermediate node $n_1,\ldots,n_m$ maintains a verifier I_{n_s} for each n_s. Initially, I_{n_s} for a particular source is set to HS_0. When n_s sends the *i*th packet, it includes HS_i with the packet.

When any intermediate node n_k receives this packet, it verifies, whether $I_{n_s} = F(HS_i)$ or not. If so, n_k validates the packet, it forwards it to the next intermediate node, and sets I_{n_s} to HS_i. In general, n_k can choose to apply the verification test iteratively up to a fixed number *w* times, checking at each step whether, $I_{n_s} = F(F\ldots(F(HS_i)))$. If the packet is not validated after the verification process has been performed *w* times, n_k simply drops the packet. By performing the verification process *w* times, up to a sequence of *w* packet losses can be tolerated, where the value of *w* depends on the average packet loss rate of the network. Here an intermediate node needs not to decrypt the packet rather it can check the authenticity of the packet before forwarding to its immediate forwarder. Figure 2 illustrates the procedure. In Figure 2(d) an adversary tries to send a bogus packet with a false HS number and it is detected in the next upstream node. Eventually such bogus packet fails to pass the authentication check and is dropped in the very next hop. This feature saves energy as the bogus packets cannot travel through the network for more than one hop.

After the tree construction, at the time of data transmission, each node could dynamically set its transmission range according to the distance of the immediate forwarder. If the distance of the forwarding node is less than the initially used transmission range for tree construction, the node decreases the range by decreasing the transmission energy. This feature gives the flexibility in our protocol to dynamically set the transmission ranges and thus helps for conserving network-wide energy. The first phase is executed every T time, where T is an application dependent parameter, which depends on the event generation rate as well as on the load of the network.

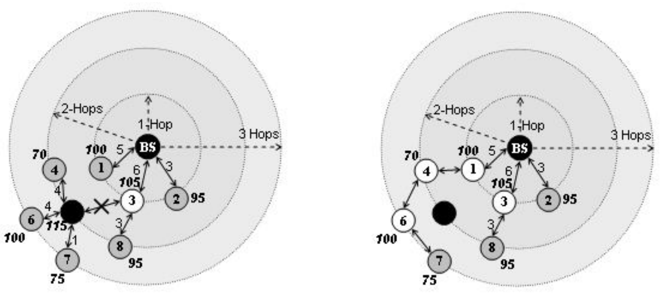

Fig. 3. (left) Node 5 failed (right) Repairing a broken path

4.3 Optional Key Refreshment

To provide data freshness and to increase the level of security, our scheme has an optional key refreshment mechanism. In this case, the base station periodically broadcasts a new session key to the sensors in the network. The format for this message is: $B|K_s|$ MAC$_{K_j}$($B|K_s$)), where, K_j is the number in the key chain number corresponding to time slot t_j. To authenticate K_s, like the OHC initialization phase, B releases the key K_j in time slot t_{j+d}. On receiving this key, the nodes can verify the integrity and source authentication of K_s. Then each node gets the new key by performing an X-OR (exclusive OR) operation with its old shared key. This method could also be utilized for refreshing the keys of a specific number of nodes. In that case, the base station could simply send the K_s to the specific node by encrypting it with its previous shared secret key. Upon receiving the new key, the node can perform the X-OR operation and could use the newly derived key for subsequent data transmissions. Changing encryption keys time-to-time has an advantage as it guarantees data freshness in the network. Moreover, it helps to maintain confidentiality of the transmitted data by preventing the use of the same secret key at all the times.

4.4 Repairing a Broken Path and OHC Re-Initialization

If in any case, any node between the source node and the BS fails, it could make one or more paths useless. Eventually, in such a case all the downstream nodes along that particular path get disconnected from the base station. To repair such a broken path, we use the stored upstream knowledge of the sensors. We know that, in the first phase each downstream node stores the ids of the one-hop upstream senders of the control message. So, this knowledge could be used for repairing the path quickly. Figure 3 shows a failed node (black) and the new path after broken path recovery.

As we are considering a highly dense deployment scenario, we believe that, in most of the cases, a node might initially get two or more upstream senders who would try to be its forwarder. This procedure works fine as long as no more than *w* packets are lost on the way, from any source node (after a path is broken due to a node failure). If within the time of repairing the path, more than *w* packets are lost from a particular source, the OHC chain along that path breaks down. In fact, this is the worst case where all the downstream nodes along the path become invalid to the base station and their sent data are discarded on the way to reach the base station. To overcome this problem, the entire OHC initialization phase could be made periodic (after certain interval). Determining the best possible time interval for re-initialization of the first phase is kept as our future work.

5 Analysis and Performance Evaluation

To understand the performance of our proposed protocol, we simulated the network using NS-2 [18] with 50 to 300 nodes uniformly distributed in a 100m×100m square sensor field. The transmission range of each sensor node was set to 25 meters. Each node was provided with 2 Joule of initial energy. Transmitter and receiver electronics were set to dissipate 50nJ/bit.m^2. The data packet length was set to 2 KB. Sink or base station was located at (150, 150) co-ordinates. We varied the number of sources from 1 to 7 and data generation interval was randomly chosen. Initially Tree construction time was set to 10 seconds. As our protocol creates a hierarchical structure in the network, we compared our protocol with two other hierarchical energy-aware routing protocols LEACH [10] and EAD [11].

Figure 4(left) shows the percentage of forwarding nodes among the total number of nodes in LEACH, EAD and our protocol. Figure 4(right) shows the energy dissipation given a number of source nodes. Less energy dissipation eventually helps for increasing the lifetime of the network. The relative gain of our proposed scheme compared to LEACH and EAD increases with the increase of number of sources. Figure 5(left) presents the number of alive nodes versus simulation time with 100 nodes. Our proposed scheme generates less number of forwarders compared to EAD. As a result, the energy dissipation is much less than that of EAD as there are less nodes participating actively in the network operation phase. Our experimental results show that, our algorithm achieves better lifetime compared to LEACH and EAD.

The method of generating and storing a long OHC in a sensor node is a little difficult. But, recently, some efficient OHC generation algorithms for resource-constrained platforms have been proposed [19], [20], [21]. Among these algorithms, the fractal graph traversal algorithm [19] could perform well on the traditional sensor nodes. This algorithm stores only some of the intermediate numbers, called pebbles, of an OHC, and uses them to compute other numbers. If the size of an OHC is n (there are total n numbers in this OHC), the algorithm performs approximately $(1/2)\log_2 n$ one-way function operations to compute the next OHC number, and requires a little more than $\log_2 n$ units of memory to save pebbles.

The length of an OHC that is needed for a source node is also an important factor. The typical length is between 2^{11} to 2^{22}. If the length of an OHC is 2^{22} and a node uses one OHC number per second, it will take more than a month to exhaust all numbers from this chain. Figure 5(right) shows the storage requirements for storing pebbles for different lengths of an OHC. This includes a skipjack based one-way function and OHC generation based on [19]. We see that a node needs about 930 bytes to maintain an OHC of length 2^{22}. This includes 256 bytes lookup table for skipjack, which can be shared with other applications. Other than this, each node has to store only a few ids and neighbor information of its one-hop neighbors. Overall, the memory requirement for our scheme could be well afforded with today's sensor nodes.

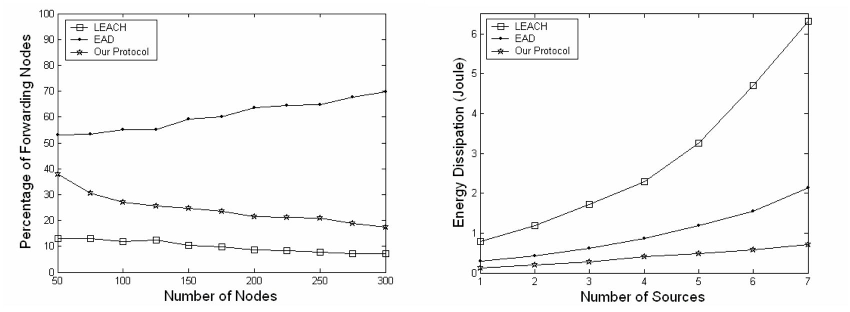

Fig. 4. (left) Percentage of forwarding nodes in total number of nodes in the network (right) Energy dissipation for different number of sources in LEACH, EAD and our protocol

We analyze the security of our scheme with respect to two design goals; the ability of BS to detect a false report, and en-route detection and filtering of false reports.

Base Station Detection. In our scheme whenever the BS receives a report from any sensor, it first checks the id of the sensor, checks the authenticity of the report by verifying the one-way hash chain number for that particular source, looks for the corresponding shared secret key and decrypts the packet. The base station could not be compromised in any way. So, it is in fact the final entity that could confirm the authenticity, confidentiality and integrity of the transmitted reports. Our security scheme is designed in a way that, any bogus report cannot reach the base station, rather would be detected and dropped by the intermediate nodes. However, if somehow a bogus packet is sent directly to the base station, it would certainly be discarded by it, for the failure of authentication check. If in any application, the optional key refreshment mechanism is employed, once the time slot of releasing the new session key is over, the base station first tries to decrypt the incoming packets from any particular source with the X-ORed new key for that node. In case, if it produces garbage result, the base station tries with the previous shared secret key with that node (the previous key could easily be obtained again by X-ORing the most recent session key with the newly computed key for that node). This case might happen when somehow some node cannot get the new session key released by the base station.

Detection by the Intermediate Nodes. (a) *Outsider Attack.* In this case, as shown in figure 2(d) that, if an outsider node generates a packet with fake OHC number, the authentication must be failed in the very next node in the path and as a result this packet would never be forwarded even to the node which is only two hops away from it. Simple verification of the OHC number prohibits the forwarding of such bogus packets. (b) *Insider Attack.* If a legitimate node along any path is compromised, the attacker could grab the OHC sequence and the shared secret key with the base station. However, it should be noticed that, to use the OHC numbers successfully, the adversary should also know the last OHC number used by that particular node to send packet to the base station. If it gets the last used OHC number, then it could use this for sending false packets successfully. Otherwise, any arbitrary use of the OHC number from that source might not be forwarded by the next intermediate node because of authentication failure. Now, in case if a node is fully compromised, that is if the adversary obtains all the required information, it actually gets the status of a legitimate node in the network. This fully compromised node could be used to generate false reports with valid authentication numbers. To prevent such type of malicious adversary, there are several factors come into play to detect the abnormal behavior of the node. In our scheme, the BS considers a report legitimate if it is reported by at least δ number of source nodes in the network, where δ is an application dependent parameter. So, the different or modified reports from a single source cannot convince the base station about any event.

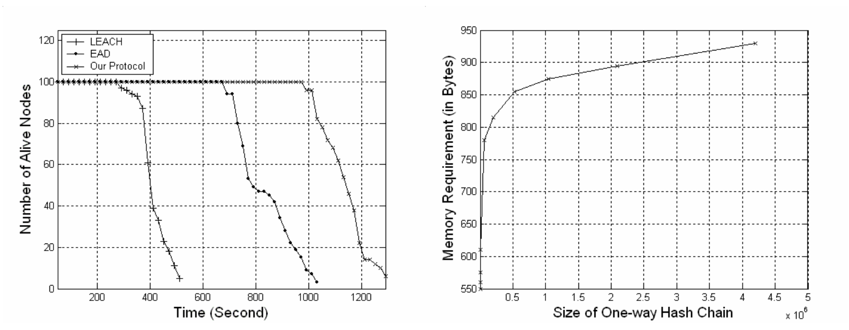

Fig. 5. (left) No. of alive nodes vs time (right) Memory requirement for OHC generation

The worst case scenario occurs if more than δ nodes in a particular region in the network are somehow compromised. This sort of collaborative and large scale attack is handled by the periodic restructuring of the whole network.

6 Conclusions

In our protocol, we mainly structured the network in a way that it could ensure the delivery of authenticated and confidential data to the BS from any source node, which is also aware of the limited energy budget of the network. According to our simulation results and analysis, our protocol demonstrates a good performance regarding energy-efficiency and security in data transmission. In future, we would like perform a detailed analysis to find out an optimal value of the interval of restructuring the network.

Acknowledgments

We would like to give special thanks to Md. Mamun-Or-Rashid for his generous help for this work.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless Sensor Net-works: A Survey. Computer Networks 38, 393–422 (2002)
2. Dai, S., Jing, X., Li, L.: Research and analysis on routing protocols for wireless sensor networks. In: Proc. of Int. Conf. on Comm. Circuits and Systems, vol. 1, pp. 407–411 (2005)

3. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: Attacks and countermeasures. Elsevier's Ad Hoc Network Journal, 293–315 (September 2003)
4. Pathan, A.-S.K., Lee, H.-W., Hong, C.S.: Security in Wireless Sensor Networks: Issues and Challenges. In: Proc. of the 8th IEEE ICACT 2006, Korea, vol. II, pp. 1043–1048 (2006)
5. Çam, H., Özdemir, S., Muthuavinashiappan, D., Nair, P.: Energy Efficient Security Protocol for Wireless Sensor Networks. Proc. of IEEE VTC 5, 2981–2984 (2003)
6. Çam, H., Özdemir, S., Nair, P., Muthuavinashiappan, D., Sanli, H.O.: Energy-efficient secure pattern based data aggregation for wireless sensor networks. Computer Communications 29(4), 446–455 (2006)
7. Ye, F., Luo, H., Lu, S., Zhang, L.: Statistical En-Route Filtering of Injected False Data in Sensor Networks. IEEE Jrnl. on Selected Areas in Communications 23(4), 839–850 (2005)
8. Zhu, S., Setia, S., Jajodia, S., Ning, P.: An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks. In: Proceedings of S&P, pp. 259–271 (2004)
9. Lee, H.Y., Cho, T.H.: Key Inheritance-Based False Data Filtering Scheme in Wireless Sensor Networks. In: Madria, S.K., Claypool, K.T., Kannan, R., Uppuluri, P., Gore, M.M. (eds.) ICDCIT 2006. LNCS, vol. 4317, pp. 116–127. Springer, Heidelberg (2006)
10. Heinzelman, W.R., Chandrakasan, A., Balakrishnan, H.: Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In: Proc. of HICSS, pp. 3005–3014 (2000)
11. Azzedine, B., Xiuzhen, C., Joseph, L.: Energy-aware data-centric routing in microsensor networks. In: Proceedings of the 8th MSWiM 03, San Diego, pp. 42–49 (2003)
12. Hyunh, T.T., Hong, C.S.: An Energy*Delay Efficient Multi-Hop Routing Scheme for Wireless Sensor Networks. IEICE Trans. on Info. & Sys. E89-D(5), 1654–1661 (2006)
13. Yin, C., Huang, S., Su, P., Gao, C.: Secure routing for large-scale wireless sensor networks. In: Proceedings of IEEE ICCT 2003, 9-11 April 2003, vol. 2, pp. 1282–1286. IEEE Computer Society Press, Los Alamitos (2003)
14. Xbow Sensor Networks: Available at: http://www.xbow.com/
15. Hass, Z.J.: Design methodologies for adaptive and multimedia networks. IEEE Communications Magazine 39(11), 106–107 (2001)
16. Heinzelman, W.B., Chandrakasan, A.P., Balakrishnan, H.: An application-specific protocol architecture for wireless microsensor networks. IEEE Transactions in Wireless Communications 1(4), 660–670 (2002)
17. Lamport, L.: Constructing digital signatures from one-way function. Technical report SRI-CSL-98, SRI International (October 1979)
18. The Network Simulator - ns-2: http://www.isi.edu/nsnam/ns/
19. Coppersmith, D., Jakobsson, M.: Almost Optimal Hash Sequence Traversal. In: 6th International Financial Cryptography 2002, Bermuda (March2002)
20. Jakobsson, M.: Fractal hash sequence representation and traversal. In: 2002 IEEE International Symposium on Information Theory, Switzerland. IEEE Computer Society Press, Los Alamitos (2002)
21. Sella, Y.: On the computation-storage trade-offs of hash chain traversal. In: The 7th International Financial Cryptography Conference, Guadeloupe (January 2003)
22. Mamun-Or-Rashid, Md., Alam, M.M., Hong, C.S.: Energy Efficient Routing for Highly Dense Sensor Networks Based on Residual Energy and Distance. In: Proc. of IEEE ICNEWS, Dhaka, Bangladesh, pp. 52–56 (2006)