Available online at www.sciencedirect.com
SCIENCE DIRECT
Computer Communications 29 (2006) 446–455
computer communications
www.elsevier.com/locate/comcom

Energy-efficient secure pattern based data aggregation for wireless sensor networks

Hasan Çam[a,*], Suat Özdemir[a], Prashant Nair[b], Devasenapathy Muthuavinashiappan[a], H. Ozgur Sanli[a]

[a]Department of Computer Science and Engineering, Ira A. Fulton School of Engineering, Arizona State University, Tempe, AZ 85287, USA
[b]Department of Electrical Engineering, Ira A. Fulton School of Engineering, Arizona State University, Tempe, AZ 85287, USA

Available online 23 February 2005

Abstract

Data aggregation in wireless sensor networks eliminates redundancy to improve bandwidth utilization and energy-efficiency of sensor nodes. This paper presents a secure energy-efficient data aggregation protocol called ESPDA (Energy-Efficient Secure Pattern based Data Aggregation). Unlike conventional data aggregation techniques, ESPDA prevents the redundant data transmission from sensor nodes to cluster-heads. If sensor nodes sense the same data, ESPDA first puts all but one of them into sleep mode and generate pattern codes to represent the characteristics of data sensed by sensor nodes. Cluster-heads implement data aggregation based on pattern codes and only distinct data in encrypted form is transmitted from sensor nodes to the base station via cluster-heads. Due to the use of pattern codes, cluster-heads do not need to know the sensor data to perform data aggregation, which allows sensor nodes to establish secure end-to-end communication links with base station. Therefore, there is no need for encryption/decryption key distribution between the cluster-heads and sensor nodes. Moreover, the use of NOVSF Block-Hopping technique improves the security by randomly changing the mapping of data blocks to NOVSF time slots. Performance evaluation shows that ESPDA outperforms conventional data aggregation methods up to 50% in bandwidth efficiency.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Sensor network; Security; Data aggregation; Pattern codes; Coverage

1. Introduction

Wireless Sensor Networks have emerged as an important new area in wireless technology. In the near future, the wireless sensor networks are expected to consist of thousands of inexpensive nodes, each having sensing capability with limited computational and communication power [1–3]. Such sensor networks are expected to be widely deployed in a vast variety of environments for commercial, civil, and military applications such as surveillance, vehicle tracking, climate and habitat monitoring, intelligence, medical, and acoustic data gathering. The key limitations of wireless sensor networks are the storage, power and processing. These limitations and the specific architecture of sensor nodes call for energy efficient and secure communication protocols. The feasibility of these inexpensive sensor networks is accelerated by the advances in MEMS (micro electromechanical systems) technology, combined with low power, low cost digital signal processors (DSPs) and radio frequency (RF) circuits [3,4].

The key challenge in sensor networks is to maximize the lifetime of sensor nodes due to the fact that it is not feasible to replace the batteries of thousands of sensor nodes. Therefore, computational operations of nodes and communication protocols must be made as energy efficient as possible. Among these protocols data transmission protocols have much more importance in terms of energy, since the energy required for data transmission takes 70% of the total energy consumption of a wireless sensor network [2]. Area coverage and data aggregation [5] techniques can greatly help conserve the scarce energy resources by

* Corresponding author. Tel.: +1 480 727 6348; fax: +1 480 965 2751.
E-mail addresses: hasan.cam@asu.edu (H. Çam), suat@asu.edu (S. Özdemir), ppn@asu.edu (P. Nair), mdeva@asu.edu (D. Muthuavinashiappan), hedo@asu.edu (H. Ozgur Sanli).

0140-3664/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2004.12.029

eliminating data redundancy and minimizing the number of data transmissions. Therefore, data aggregation methods in sensor networks are extensively investigated in the literature [5–8].

In SPIN (Sensor Protocols for Information via Negotiation) [9], the transmission of redundant data is eliminated using the meta-data negotiations by nodes. In directed diffusion, gradients are set up to collect data and data aggregation makes use of positive and negative reinforcement of paths [7]. In [8] sensor nodes send pattern of data, which shows how sensor's readings change over a predefined time interval. Cluster-head collects data patterns and sends only the ones matching critical events, such as sudden temperature drop forecasting a storm, to the base station. Cluster-head can also send k representative readings instead of n readings obtained from all of its sensors, according to the k-means algorithm.

Security in data communication is another important issue to be considered while designing wireless sensor networks, as wireless sensor networks may be deployed in hostile areas such as battlefields [2,10,11]. Therefore, data aggregation protocols should work with the data communication security protocols, as any conflict between these protocols might create loopholes in network security.

This paper presents an Energy-efficient and Secure Pattern based Data Aggregation (ESPDA) protocol which considers both data aggregation and security concepts together in cluster-based wireless sensor networks. Although data aggregation and security in wireless sensor networks have been studied in the literature, to the best of our knowledge this paper is the first study to consider data aggregation techniques without compromising security. ESPDA uses *pattern codes* to perform data aggregation. The pattern codes are basically representative data items that are extracted from the actual data in such a way that every pattern code has certain characteristics of the corresponding actual data. The extraction process may vary depending on the type of the actual data. For example, when the actual data are images of human beings sensed by the surveillance sensors, the key parameter values for the face and body recognition are considered as the representative data depending on the application requirements. When a sensor node consists of multiple sensing units, the pattern codes of the sensor node are obtained by combining the pattern codes of the individual sensing units. Instead of transmitting the whole sensed data, sensor nodes first generate and then send the pattern codes to cluster-heads. Cluster-heads determine the distinct pattern codes and then request only one sensor node to send the actual data for each distinct pattern code. This approach makes ESPDA both energy and bandwidth efficient. ESPDA is also secure because cluster-heads do not need to decrypt the data for data aggregation and no encryption/decryption key is broadcast. Additionally, the proposed nonblocking OVSF block hopping technique further improves the security of ESPDA by randomly changing the mapping of data blocks to NOVSF time slots. The preliminary versions of the proposed security protocol appear in [10,11].

Sensor nodes are usually deployed in high density in order to cope with node failures due to harsh environments, yielding highly overlapping sensing regions of the nodes. Random deployment of the network also results in many areas to be covered by more than one sensor node. Therefore, it is highly desirable to ensure that an area is covered by only one sensor node at any time, so that no more than one sensor node senses the same data. This leads to an improvement for the efficiency of data aggregation since the redundant data is not even sensed. In this regard, this paper introduces an algorithm to coordinate the sleep and active modes when sensor nodes have overlapping sensing ranges.

The rest of the paper is organized as follows. Section 2 describes the data aggregation and sleep-active mode protocol. Section 3 introduces the security protocol. Section 4 presents the performance evaluation of the proposed data aggregation, sleep-active, and security protocols. Concluding remarks are made in Section 5.

2. Data aggregation in ESPDA

This paper considers sensor networks with hierarchical structure where data is routed from sensor nodes to the base station through cluster heads. Base stations are assumed to have sufficient power and memory to communicate securely with all the sensor nodes and external networks such as Internet. Sensor nodes are deployed randomly over an area to be monitored and organize themselves into clusters after the initial deployment. A cluster-head is chosen from each cluster to handle the communication between the cluster nodes and the base station. Cluster-heads are changed dynamically based on residual energy in order to have uniform power consumption among all sensor nodes.

Since data transmission is a major cause of energy consumption, ESPDA first reduces transmission of redundant data from sensor nodes to cluster-heads with the help of sleep-active mode coordination protocol. Then, data aggregation is used to eliminate redundancy and to minimize the number of transmissions for saving energy. In conventional data aggregation methods, cluster-heads receive all the data from sensor nodes and then eliminate the redundancy by checking the contents of the sensor data. ESPDA uses pattern codes instead of sensed data to perform data aggregation; therefore, the contents of the transmitted data do not have to be disclosed at the cluster-heads. This enables ESPDA to work in conjunction with the security protocol. In security protocol, sensor data, which is identified as non-redundant by the cluster-heads, is transmitted to the base station in encrypted form. The pattern codes are generated using a secret pattern seed broadcast by the cluster-head periodically. Pattern seed is a random number used for improving the confidentiality of the pattern codes by not allowing

the same pattern codes produced all the time. As the pattern seed is changed, pattern generation algorithm produces a different pattern code for the same sensor data. Thus, redundancy is eliminated even before the sensor data is transmitted from the sensor nodes.

2.1. Sleep-active mode coordination

Each sensor node is set to either idle or active mode for sensing operation based on the connectivity and conditions of the sensing environment. One technique is to focus on reducing redundant data to be transmitted from sensor nodes to cluster-heads. In ESPDA, redundant packets are already eliminated at the cluster-head level, but it is still possible to conserve energy by exploiting the data driven communication nature of the wireless sensor network. Identifying nodes that have overlapping sensing ranges and turning off the sensing units of some of those nodes for a bounded amount of time reduces energy wastage since these nodes will produce redundant data due to the overlapping. Most radio energy models in the literature emphasize the communication aspect of the wireless sensor nodes. It is possible to argue that additional gains achieved by data driven sleep coordination method may not be significant. However, the benefits of this technique can be observed if one focuses on operation of a sensor network. Consider a case in which a node's sensing area is in the union of its neighbors' sensing regions. Whenever an event occurs in its sensing range, it will send the corresponding pattern code to its cluster-head among with its neighbors. Since its packet will eventually be eliminated at the cluster-head, this is a redundant transmission. Note that this is the best-case scenario since all its neighbors transmit to the same cluster-head. It is possible that this node reports to a different cluster-head other than its neighbors due to the clustering scheme itself. The key point to conclude from this observation is that, failure to eliminate a redundant transmission at the originating sensor node may result in a number of unnecessary transmissions in the clustering hierarchy, which may only be eliminated at the higher-level cluster-heads or at the base station in the worst case.

In the sleep protocol a sensor node cooperates with its neighbors to identify the overlapping coverage regions. Neighboring nodes can communicate with each other via cluster-head. The term *sleep protocol* is used to refer turning off the sensing unit of the nodes rather than turning off the radio. The total lifetime of the network is divided into fixed length slots of duration T as shown in Fig. 1. Each slot consists of *observation*, *learning* and *decision* phases. Each node that is awake updates its local buffer based on the events in its observation phase. In the learning phase, nodes exchange summary of their buffer contents, such as set of hash values of events observed, with their neighbors. In the decision phase, each node evaluates its eligibility to keep its sensing unit on/off for a duration Z' (multiple of T) and broadcasts its decision to its neighbors. The order of

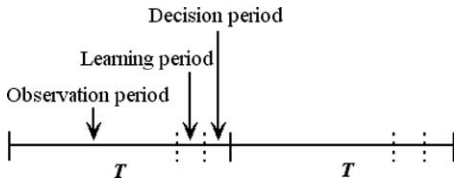

Fig. 1. Representation of a time slot T consisting of observation, learning and decision phases.

broadcasts for nodes is arranged as $(Z/Z_{max}) * B$, where B is a constant smaller than T, Z (multiple of T) is the previous sleep duration of the node and Z_{max} (multiple of T) is the maximum duration that a node can sleep respectively. The motivation behind the ordering of broadcasts is to reduce the conflicts and prioritize the nodes, which were not able to sleep in the previous decision phases. When its timer expires, each node performs the following algorithm.

Algorithm. OnBroadcastTimerExpire()

Begin

1. if (events in buffer observed by neighbors which has not broadcast their decision yet)
2. $Z' = \min(2 * Z, Z_{max})$
3. Broadcast sleep decision to neighbors
4. Turn off sensing unit for duration Z
5. else
6. $Z' = 0.5 * T$
7. Stay awake for next slot
8. endif
9. Flush event buffer
End

The arrangement of sleep times is similar to the binary exponential back off algorithm. At the end of its sleep time, each node stays awake for one slot duration to base its decision criteria on the recent observations, changes in the network and the environment (Fig. 2). The performance evaluation of the sleep protocol is given in Section 4.1.

2.2. Data aggregation using pattern codes

The pattern generation (PG) algorithm is executed on all sensor nodes to generate the pattern codes specific to the sensed data. In the algorithm sensor data are mapped to a set of numbers, the range of numbers is divided into intervals such that the boundaries and width of intervals are determined by the predefined threshold values. The number of threshold values and the variation of intervals may depend on the user requirements and *precision* defined for the environment in which the network is deployed. The pattern generation algorithm then computes the *critical values* for each interval using the pattern seed and generates the lookup tables, *interval* and *criticalvalue*. The *interval* lookup table defines the range of each interval and the

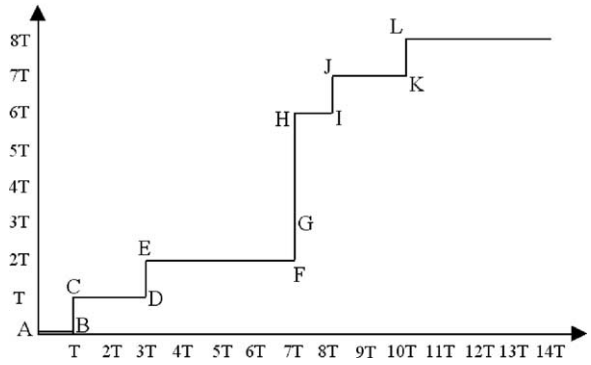

Fig. 2. Representation of the sleep protocol: Node X is initially put to sleep mode at point A and it sleeps for a period of time (T) until point B. During the period between B and C, node X observes the environment and inserts the pattern codes into its buffer, then exchanges its buffer contents with active neighboring sensors. At point C, there is a match between buffer contents, so node X again turns off its sensing unit and goes to sleep for $2T$ until point D. When buffer contents do not match, like in point G, then node X has to stay active until one of its neighbors wakes up and node X has the right to go to sleep mode (point H).

criticalvalue lookup table maps each interval to a critical value. For example, the critical values may be assigned as 1 for the first interval and vary through 9 being the last interval. These critical values form the base for generation of pattern codes.

The characteristics of sensor data are represented by parameters such as temperature or humidity. When data is sensed from the environment, its parameters are compared with the intervals defined in the lookup table of PG algorithm and a corresponding critical value is assigned to each parameter. Concatenating the critical values of all parameters of the data generates the pattern code. The timestamp and the sensor ID are appended to pattern codes when they are transmitted to cluster-head. The pattern comparison algorithm at the cluster-head eliminates the redundant pattern codes, which in turn prevents redundant transmission of data. Sensor nodes send the set of encrypted data, which has no redundancy to the base station via cluster-head.

In the pattern generation algorithm, the pattern seed is used to generate pattern codes. When the clusters are initially established in the network, sensor nodes receive the secret pattern seed from their corresponding cluster-heads where the pattern seed is a random number generated and broadcast by the cluster-head in encrypted format. The details of how the pattern seed is broadcasted securely is presented in Section 3.1. The pattern seed is used for improving the confidentiality of the pattern codes by not allowing the same pattern codes produced all the time. As the pattern seed is changed, pattern generation algorithm produces a different pattern code for the same data. Therefore, pattern seed is changed at regular time intervals.

Algorithm. Pattern generation (PG)

Input: Sensor reading D.

Data parameters being sensed.

threshold[] : Array of threshold levels of data intervals for each data type.

Data precision requirements of the network for each data parameter.

S(critical[], seed*)*: Function to shuffle the mapping of critical values to data intervals.

Output: Pattern-code (**PC**)

Begin

1. **Variable** PC = []; // Initialize the pattern code
2. **for** each data parameter
3. **Declare** n; //Number of data intervals for this data type
4. **Declare** *interval*[n], *criticalvalue*[n]; // Lookup tables
5. Extract the data from D for the corresponding data parameter.
6. Round off data for the precision required by the corresponding data parameter.
7. **for** i = 1 to n
8. *interval*[i] = *threshold*[i-1] – *threshold*[i] ;
9. **endfor**
10. **if** (new seed sent by cluster-head) **then**
11. // Refresh the mapping of critical values to data intervals
12. **for** i = 1 to n
13. *criticalvalue*[i] = *S*(*criticalvalue*[i], seed) ;
14. **endfor**
15. **endif**
16. Find the respective critical value for each current data sensed using *interval* and *criticalvalue* lookup tables.
17. PC = PC + [critical value] ; // Concatenate critical value to pattern code
18. **endfor ;**
19. PC = PC + [Timestamp] + [Sensor Id] ; // Append timestamp and sensor id

End

2.2.1. Pattern generation example

Let D ($d1$, $d2$, $d3$) denote the sensed data with three parameters $d1$, $d2$, and $d3$ representing temperature, pressure and humidity, respectively, in a given environment. Each parameter sensed is assumed to have threshold values between the ranges 0 to 100 as shown in Table 1. The pattern generation algorithm performs the following steps

- Pattern code to be generated is initialized to empty pattern code (line 1).

Table 1
Look up tables for data intervals and critical values

Threshold	30	50	70	80	90	95	100
Interval	0–30	31–50	51–70	71–80	81–90	91–95	96–100
Critical	5	3	7	8	1	4	6

Table 2
Pattern codes generation table

	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5
Data	D(56, 92, 70)	D(70, 25, 25)	D(58, 93, 69)	D(68, 28, 30)	D(63, 24, 26)
D1 critical val.	7	7	7	7	7
D2 critical val.	4	5	4	5	5
D3 critical val.	7	5	7	5	5
Pattern code	747	755	747	755	755

- The algorithm iterates over sensor reading values for parameters of data that are being sensed. In this case, it first considers temperature (line 2).
- Temperature parameter is extracted from sensor reading D (line 5).
- For the temperature parameter, the algorithm first checks whether a new seed is received from the cluster-head (line 10). Arrival of a seed refreshes the mapping of critical values to data intervals. As an example, the configuration in Table 1 can be reshuffled as {8, 3, 5, 1, 7, 6, 4}, changing the critical value for the first interval ([0–30]) from 5 to 8 while keeping the second interval ([31–50])'s critical value as 3, same with the previous case.
- The data interval that contains the sensed temperature is found from the *interval* table. Then, from the interval value, corresponding critical value is determined from *criticalvalue* table (line 16). Table 2 shows the critical values for different sensor readings if the same lookup tables of Table 1 are used for temperature, pressure and humidity.
- PC is set to the new critical value found (line 17). For the pressure and humidity, corresponding critical values are appended to the end of partially formed PC.
- Previous steps are applied for the pressure and humidity readings
- When full pattern code is generated, timestamp and sensor identifier is sent with the pattern code to the cluster-head (line 19).

In this example, data sensed by sensor 1 and sensor 3 are same with each other as determined from the comparison of their pattern code values, 747. Similarly, data sensed by sensor 2, sensor 4 and sensor 5 are the same (pattern code value 755). Pattern codes with the same value are referred as a *redundant set*. Hence, the cluster-head selects only sensor 1 and sensor 4 to transmit the data from each redundant set based on the timestamps according to the Pattern Comparison Algorithm described below.

2.3. Pattern comparison by cluster-head

The cluster-head waits for sensor nodes to transmit the pattern codes. After receiving pattern codes from the sensor nodes for a period T, the entire set of codes is classified based on redundancy. The period T varies based on the environment where the sensor network is deployed. Unique patterns are then moved to the 'selected-set' of codes. The sensors nodes that correspond to the unique pattern set ('selected-set') are then requested to transmit the actual data. ACK signals may be broadcast to other sensors ('de-selected-set') to discard their (redundant) data. These sensor nodes can be put to sleep mode to conserve power.

Algorithm. Pattern Comparison

Input: Pattern codes

Output: Request sensor nodes in the selected-set to send actual encrypted data.

Begin

1. Broadcast 'current-seed' to all sensor nodes
2. **while** (current-seed is not expired)
3. time-counter = 0
4. **while** (time-counter < T)
5. get pattern code, sensor ID, timestamp
6. **endwhile**
7. Compare and classify pattern codes based on redundancy to form 'classified-set'.
8. selected-set={one pattern code from each classified-set}
9. deselected-set = classified-set – selected-set
10. **if** (sensor node is in selected-set)
11. Request sensor node to send actual data
12. **endif**
13. **endwhile**
End

3. Security protocol of ESPDA

In sensor network applications the security aspect is as important as performance and energy consumption of the network. The security protocols used in the sensor networks must be carefully designed considering the energy and computational resource constraints of wireless sensor networks. Asymmetric cryptographic algorithms are not suitable for providing security on wireless sensor networks due to the limited computation, power, and storage resources available on wireless sensor nodes. Therefore, symmetric key cryptographic algorithms are employed to support security in wireless sensor networks [2]. In this section, the energy-efficient security protocol of ESPDA is presented.

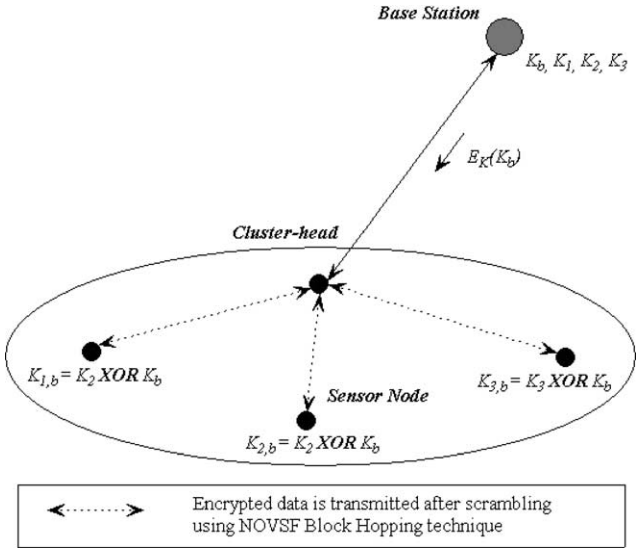

Fig. 3. A cluster of sensors and base station computing encryption keys, $K_{i,b}$. Sensor computes $K_{i,b}$ by XORing K_i with K_b that is broadcasted by base station.

Each sensor node is assigned a unique (ID_i), a node specific secret key (K_i), and a secret key (K) common to all nodes. K_i, K, and ID_i are assigned during the manufacturing phase. Before the deployment of network, base station is provided with K and all (ID_i-K_i) pairs used in the network.

The cluster-heads use common secret key, K, to transmit the pattern seed securely to cluster members to be used in the pattern generation algorithm. In addition, base station generates a session key (K_b) for each session and broadcasts to all sensor nodes in encrypted format, $E_K(K_b)$. A session is a certain amount of time period whose duration depends on the security requirements of the application. Whenever base station broadcasts a new K_b, all the sensor nodes generate their secret session keys ($K_{i,b}$) by XORing the K_b with K_i. Since the sensor nodes have built-in keys, the protocol avoids the distribution of secret keys in the wireless environment (Fig. 3).

3.1. Security algorithms

The security protocol of ESPDA consists of two security algorithms, namely Security Algorithm at Sensor Node (SAS) and Security Algorithm at Base Station (SAB). SAS and SAB employ symmetric key encryption to provide data integrity, confidentiality and authenticity.

SAS (Implemented in Sensor Nodes)
Step 1: If sensor node i wants to send data to its cluster head, go to next step, otherwise exit the algorithm.
Step 2: Sensor node i requests the cluster head to send the current session key K_b. This session key is broadcasted in an encrypted format using the common encryption key K, $E_K(K_b)$.
Step 3: Sensor node i XORs the current session key K_b with its built-in key K_i to compute the encryption key $K_{i,b}$.
Step 4: Sensor node i encrypts the data with $K_{i,b,}$ and appends its ID_i, the time stamp and MAC($K_{i,b}$, Data) to the encrypted data and then sends them to the cluster head using NOVSF-BH technique.
Step 5: Cluster head receives the data, appends its own IDi, and then sends them to the higher-level cluster head or the base station. Go to step 1.

SAB (Implemented in Base Station)
Step 1: If there is any need to broadcast the session key K_b broadcast $E_K(K_b)$ to all sensor nodes.
Step 2: If there is no data being sent to the base station go to step 1 after the session is complete.
Step 3: Compute the encryption key, $K_{i,b,}$ using the IDi of the sensor node and the time stamp within the data. Base station then uses the $K_{i,b}$ to decrypt the data.
Step 4: Check the time stamp and $K_{i,b}$ for the data freshness and calculate MAC($K_{i,b}$, Data) to verify the data integrity. If the data is altered or replayed, then discard the data and go to Step 6.
Step 5: Decide whether to request all sensor nodes for retransmission of data. If not necessary then go to Step 1.
Step 6: Ask corresponding sensor nodes to transmit their data again. When the session is finished, generate a new session key and go to Step 1.

In each session base station broadcasts a new session key K_b, encrypted using the common encryption key K, i.e. $E_K(K_b)$ (Step 1 of SAB). Sensor nodes receive broadcasted session key K_b and compute their node-specific secret session encryption key ($K_{i,b}$) by XORing the K_b with Ki (Step 3 of SAS). Data confidentiality is provided by using $K_{i,b}$ for all the consequent data encryption and decryption during that session by both algorithms. Since each sensor node calculates $K_{i,b}$ using its unique built in key, encrypting data with $K_{i,b}$ also provides data authentication in the proposed algorithm. Changing encryption keys in each session guarantees data freshness in the sensor network, moreover, it also helps to maintain the confidentiality of the transmitted data by preventing the use of the same secret key at all times. Ensuring data freshness means that no adversary replayed old messages and data is recent. During the data transmission, each sensor node appends its ID_i, time stamp and message authentication code, MAC($K_{i,b}$, Data), to the messages to verify data freshness and integrity (Step 4 of SAS). Upon receiving a message, base station finds out K_i associated with the ID_i on the message, and then decrypts the data using $K_{i,b}$ (Step 3 of SAB). SAS and SAB use the Blowfish [12] to encrypt and decrypt the data. The evaluation of Blowfish and some other encryption algorithms are given in Section 4.3. In addition to using Blowfish for data confidentialty, the proposed security protocol offers additional security by sending the encrypted data using the NOVSF-BH technique (Step 4 of SAS). The details of NOVSF-BH technique are explained in Section 3.2. Upon receiving the data from the sensor nodes, the cluster head appends its own ID_i before forwarding the data to the base station (Step 5 of SAS). Appending cluster-head ID_i to each sensor data helps the base station in locating the origin of the sensor data and reduces the search time required to find the K_i associated with originating node ID_i (Step 3 of SAB).

When the base station receives sensor data, it first determines the K_i of the sender node by using sender and cluster-head ID_is, and computes the $K_{i,b}$ to decrypt the data (Step 3 of SAB). Then, base station checks the time stamp and $K_{i,b}$ for the data freshness and calculates MAC($K_{i,b}$, Data) to verify the data integrity (Step 4 of SAB). If the data is altered or replayed, then base station requires corresponding sensor nodes to transmit their data again (Step 6 of SAB). In both SAS and SAB algorithms, message authentication codes are calculated using memory efficient CBC-MAC [13] protocol. In CBC-MAC protocol, message authentication codes are actually calculated by the encryption cipher without any need for additional message authentication algorithm.

3.2. NOVSF Block hopping technique

The third generation (3G) wireless standards UMTS/IMT-2000 use the wideband CDMA (WCDMA) to support high rate and variable bit rate services with different quality of service (QoS) requirements. In the 3GPP specifications, orthogonal variable spreading factor (OVSF) codes [14] are used as the channelization codes for data spreading for both downlink and uplink. NOVSF codes are Nonblocking in the sense that no code assignment blocks the assignment of any other code. All NOVSF codes are orthogonal to each other and, therefore, can be assigned simultaneously as far as orthogonality is concerned [15,16].

Due to resource constraints, the biggest challenge in sensor network security protocols is the implementation of cryptographic primitives. This challenge might require sacrificing some security to achieve feasibility; like using small keys or weak cryptographic algorithms. Hence, to provide complete security, we introduce the NOVSF Block-Hopping (NOVSF-BH) technique without utilizing additional power for implementation. In this paper, SAS uses NOVSF-BH technique to improve security and spectral efficiency of wireless sensor networks. In NOVSF codes, each OVSF code has 64 time slots such that any number of these timeslots can be assigned to a channel. The proposed NOVSF-BH technique takes advantage of these time slots by assigning data blocks to time slots using different mappings in every session. Base station periodically sends out different mapping permutations to cluster heads by encrypting them with the common encryption key K. An example of mapping data blocks to time slots is shown in Fig. 4.

In SAS, before sensor nodes send data to base station, they first encrypt data as explained in Section 3.1, and then apply the NOVSF-BH technique. By NOVSF-BH the malicious user would have to first find the mapping pattern for that particular session and then try decrypting the data using the secret symmetric key, which makes the data transmission more secure. The advantage of this technique is that it increases communication security without requiring additional energy. The additional mapping of data blocks to NOVSF time slots is accomplished by adding a multiplexer to the system.

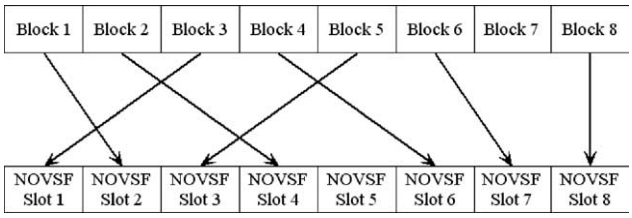

Fig. 4. NOVSF Block-hopping Technique: Mapping data blocks to NOVSF time slots where data blocks are transmitted in the following order in time: block 3, block 1, block 5, block 2, block 7, block 4, block 6, and block 8, assuming that these eight blocks are available in buffer.

4. Performance evaluation

This section presents the performance evaluation of ESPDA proposed in the paper using GloMoSim [17] and our own simulators written in C language. We first start with the performance evaluation results of sleep protocol in Section 4.1, followed by energy efficiency of pattern based data aggregation in Section 4.2 and evaluation of security protocol algorithms in Section 4.3.

4.1. Sleep-active mode protocol

In this section, the performance of the sleep-active mode protocol is evaluated. The target area to be monitored is a 100 m × 100 m square region. The base station is placed at coordinate (0, 0). The results for each variable are averaged over 10 iterations for a specific value for the variable. Each instance of the network is connected and provides full coverage initially. Event buffer size of the nodes is 5, while the maximum windows size for sleep duration of the nodes is set to 8. The values for number of nodes, transmission range and number of events per second inserted to the network are 200, 20 m and 10, respectively. The time values are represented in terms of slot time T. At each slot, 10 events are inserted at random locations in the network. The resolution of the grid for placement of the nodes and events is 1 m. Following figures illustrate the effect of increasing sensing range of the nodes on energy consumption of the network and number of undetected events due to the blind spots that may occur because of the sleeping protocol with 1-byte patterns. As can be observed from the Fig. 5 increasing the sensing range of the nodes enables nodes to sleep longer yielding higher energy savings. Similarly, Fig. 6 shows that increasing the sensing range enhances the coverage of the network yielding less percentage of the events to go undetected when the sleep protocol is used.

The energy gain illustrated in Fig. 5 is a lower bound on the actual savings of the protocol since only transmissions between clusterheads and sensor nodes are considered.

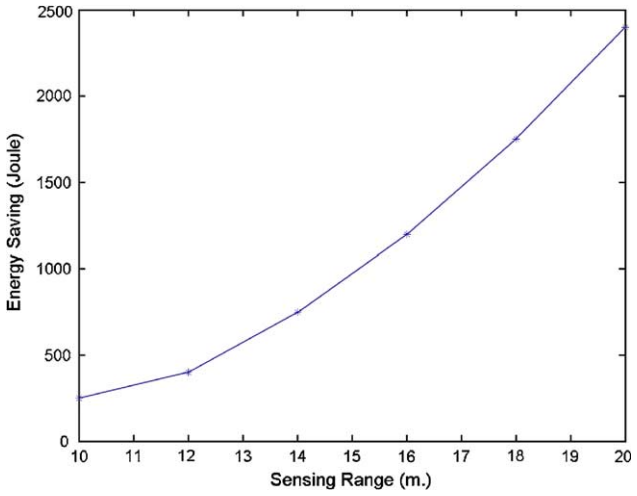

Fig. 5. Energy efficiency due to sleep protocol.

Remaining nodes also have higher coverage due to their increased sensing range, which reduces percentage of undetected events.

4.2. Energy efficiency of pattern based data aggregation

ESPDA is based on pattern codes, which are not used in the conventional data aggregation algorithms. In what follows, the system delay is computed first.

Let

- R denote the transmission rate at which sensors can send data (bps),
- N denote the total number of active sensor nodes, M of which have distinct data,
- D_i denote the number of bits transmitted per session by sensor node i,
- P_i denote the number of pattern code bits transmitted per session by sensor node i,

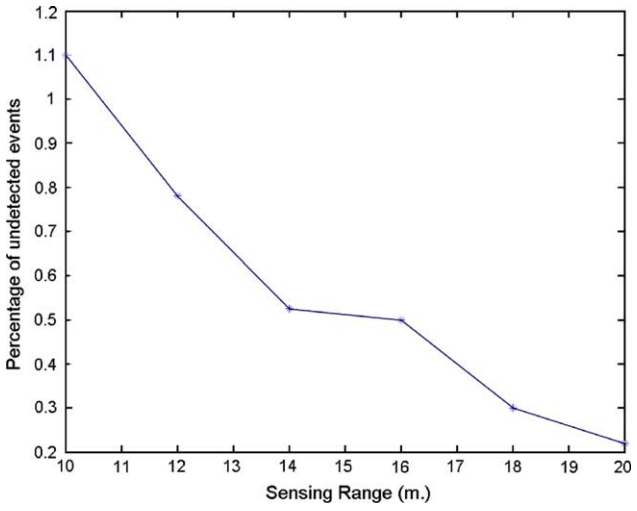

Fig. 6. Blind spot effect of the protocol on coverage of the network.

Assuming that the pattern generation time as well as the propagation delay between sensor nodes and cluster-head are negligible, the data transfer time from sensor nodes to cluster-head is computed below by including the data transmission time only. Hence, in case of the conventional data aggregation algorithm, the data transfer time equals:

$$T_{\text{CONVENTIONAL\_DA}} = \sum_{i=1}^{N} (D_i/R).$$

However, in case of ESPDA, all N sensor nodes first send pattern codes and, then, only M of them send data, thereby resulting in the following data transfer time:

$$T_{\text{ESPDA}} = \sum_{i=1}^{N} (P_i/R) + \sum_{i=1}^{M} (D_i/R).$$

Next, we show how much less data is transmitted in case of ESPDA compared to a conventional data aggregation algorithm. The amount of data to be transmitted by a conventional data aggregation algorithm can be expressed as $\sum_{i=1}^{N} D_i$.

However, in case of ESPDA, the amount of transmitted data is written:

$$\sum_{i=1}^{M} D_i + \sum_{i=1}^{N} P_i$$

where M equals the number of those sensor nodes that have distinct data, which implies that M is usually much less than N. In addition, P_i is expected to be a very small percentage of D_i. Consequently, ESPDA usually requires much less bandwidth than the conventional data aggregation algorithm, depending on the redundancy rate.

To assess the energy efficiency of ESPDA, we wrote a simulator in C and used GloMoSim. Our simulator is used in Pattern Generation and Pattern Comparison aspects of ESPDA. GloMoSim is used to simulate the transmission of data and pattern codes from sensor nodes to cluster-head. Simulation results show that ESPDA improves energy efficiency significantly by reducing the number of packets transmitted in data communication as shown in Fig. 7.

In our simulations we have considered the communication between the sensor nodes and base station. Bandwidth occupancy rate is defined as the ratio of the bandwidth used by data aggregation to the bandwidth used without any data aggregation. Compared to conventional data aggregation algorithms, as the redundancy increases the bandwidth efficiency of ESPDA also increases (Fig. 7). At 100% redundancy, the bandwidth occupancy of ESPDA approaches to zero, since ESPDA eliminates redundancy before sensor nodes transmit the actual data packets. However, in conventional data aggregation bandwidth occupancy is more than 50% of the total bandwidth since all sensor nodes transmits the actual data to cluster-head for aggregation.

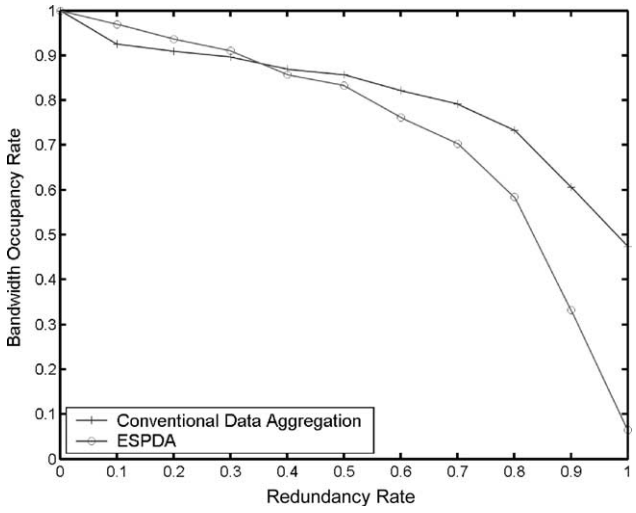

Fig. 7. Occupied bandwidth rate versus redundancy rate for ESPDA and conventional data aggregation.

4.3. Performance evaluation of ESPDA security protocol

This section presents the evaluation of several cryptographic algorithms that can be used in ESPDA. Considering the limited resources of sensor nodes, the cryptographic algorithms used in sensor networks must be chosen carefully in terms of their code size and energy consumption. For example, a SmartDust [18] node has only 8-bit, 4 MHz processor with a 916 MHz radio. The algorithms evaluated in this paper are AES [19], RC5 [20], DES [21] and Blowfish [12,22]. The evaluation metrics are energy efficiency and time. All of the evaluated algorithms are proven to be cryptographically secure [19,23]; therefore cryptanalytic strength of these algorithms is not included in the evaluation metrics.

First, algorithms are evaluated in terms of memory space. AES uses over 800 bytes of look up tables, which is too large for memory space limited sensor nodes. Similarly, DES block-cipher, which needs a 512-entry SBOX table and a 256-entry table for various permutations, requires too large memory space for computations. RC5 is also shown to be too large to fit in a sensor node [2]. With modifications described in [12], Blowfish is the most suitable algorithm for sensor nodes in terms of memory requirements. Blowfish requires 1 KB memory for encryption and 400-Byte for key setup. CBC-MAC also requires 580-Byte in the smallest case [2]. Hence, the total memory space of Blowfish is around 2 KB (Table 3).

Then, all of the algorithms are simulated on *JouleTrack* energy simulator [24] in order to attain their energy consumption when executing on StrongARM 1100 processor. The energy estimation model of JouleTrack is based on energy consumption of the instructions of ARM instruction set. Experiments of this energy model show accuracy within %3 of actual energy consumption [24]. The simulation results for AES, RC5, DES, and Blowfish is presented in Table 4 and Fig. 8. Table 4 shows the time required to encrypt and decrypt 32 Byte data and throughput of each cryptographic algorithm. Fig. 8 presents the graphical comparison of the cryptographic algorithms based on their execution time and energy consumption on *JouleTrack*. As seen from Fig. 8 and Table 4, Blowfish and AES clearly outperform RC5 and DES. Even though Blowfish requires %15 more execution time and energy consumption than AES, memory space requirement of Blowfish is much less than AES. Therefore, Blowfish is the most suitable cryptography algorithm for our security protocol.

Table 3
Memory space consumption (in bytes)

Encryption	MAC operations	Key setup	Total memory consumption
1000	580	400	2080

Table 4
Our simulation results for cryptographic algorithms

Algorithm	Key length (bit)	Total time for 32B data input (Microsecond)	Through-put (Kbps)
RC5	128	3391	75.4939
AES	128	2070	123.6714
DES	56	3822	68.9806
Blowfish	128	2444	104.7463

5. Conclusion

This paper has introduced an energy-efficient security algorithm and data aggregation protocol, which together forms the ESPDA. In contrast to conventional data aggregation protocols, ESPDA avoids the transmission of redundant data from the sensor nodes to the cluster-head. In ESPDA, cluster-heads are not required to examine the data received from the sensor nodes. Instead the cluster-heads compare the pattern codes received from the sensor nodes

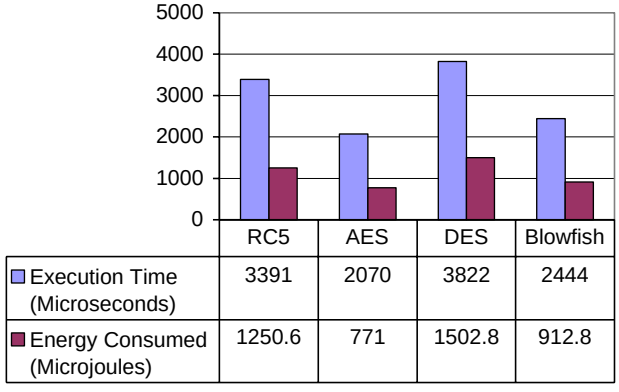

Fig. 8. Execution time required by each cryptographic algorithm for 32-byte output and energy consumption of each cryptographic algorithm to encrypt 32-byte data [23].

and decide on which sensor nodes need to transmit the sensor data. This leads to better bandwidth utilization and energy efficiency since no redundant actual data is transmitted from sensor nodes to cluster-head. The sleep-active mode protocol is introduced in ESPDA, which reduces the number of active sensor nodes depending on their overlapping sensing range. We have also presented a security protocol and NOVSF Block-Hopping technique that provides data communication security. One of the important factors is that the symmetric keys used in the security algorithms are not transmitted, between the cluster-head and the sensor nodes.

Simulation results show that as the redundancy rate increases, ESPDA bandwidth occupancy decreases thereby improving its bandwidth efficiency. As the density of network increases, the energy saved by using the sleep-active mode protocol also increases. This also enhances the overall energy and bandwidth efficiency. From the security point of view, our simulation results prove that cryptographic primitives that are used in our security protocol give shorter execution time and higher throughput using less energy as compared to other cryptographic primitives. As for the future work, we will focus on feature extraction methods for pattern generation.

References

[1] W. Ye, J. Heidemann, D. Estrin, An energy-efficient MAC protocol for wireless sensor networks, Proceedings of INFOCOM 2002, vol. 3 2002 pp. 1567–1576.

[2] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, D.E. Culler, SPINS: Security protocols for sensor network, Wireless Networks 8 (5) (2002) pp. 521–534.

[3] A. Sinha, A. Chandrakasan, Dynamic power management in wireless sensor networks, IEEE Design and Test of Computers 18 (2) (2001) pp. 62–74.

[4] A. Chandrakasan, R. Amirtharajah, S. Cho, J. Goodman, Design considerations for distributed micro sensor systems, Proceedings of IEEE Customs Integrated Circuits Conference (CICC), May, 1999 pp. 279–286.

[5] C. Intanagonwiwat, D. Estrin, R. Govindan, J. Heidemann, Impact of network density on data aggregation in wireless sensor networks, Proceedings of the 22nd International Conference on Distributed Computing Systems, July, 2002 pp. 575–578.

[6] E. Cayirci, Data aggregation and dilution by modulus addressing in wireless sensor networks, IEEE Communications Letters 7 (8) (2003) pp. 355–357.

[7] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed Diffusion for Wireless Sensor Networking, IEEE/ACM Transactions on Networking 11 (1) (2002) pp. 2–16.

[8] M. Khan, B. Bhargava, S. Agarwal, L. Lilien, Pankaj, Self-configuring Node clusters, data Aggregation and Security in Micro sensor Networks, http://raidlab.cs.purdue.edu/papers/self-config.pdf.

[9] W.R. Heinzelman, J. Kulik, H. Balakrishnan, Adaptive protocols for information dissemination in wireless sensor networks, Proceedings of the 5th Annual ACM/IEEE International Conf. on Mobile Computing and Networking, 1999 pp. 174–185.

[10] H. Çam, S. Özdemir, D. Muthuavinashiappan, Prashant Nair, Energy-efficient security protocol for wireless sensor networks, IEEE VTC Fall 2003 Conference, October, 2003 pp. 2981–2984.

[11] H. Çam, S. Özdemir, Prashant Nair, D. Muthuavinashiappan, ESPDA: energy-efficient and secure pattern-based data aggregation for wireless sensor networks, IEEE Sensors—The Second IEEE Conference on Sensors, October, 2003 pp. 732–736.

[12] B. Schneier, Fast Software Encryption, Cambridge Security Workshop Proceedings, Springer, Berlin, 1994. pp. 191–204.

[13] National Institute of Standards and Technology (NIST), DES model of operation, Federal Information Processing Standards Publication 81 (FIPS PUB 81) (1981).

[14] F. Adachi, M. Sawahashi, K. Okawa, Tree-structured generation of orthogonal spreading codes with different length for forward link of DS-CDMA mobile radio, Electronic Letters 33 (1) (1997) 27–28.

[15] H. Cam, Nonblocking OVSF codes and enhancing network capacity for 3G wireless and beyond systems, Special Issue of Computer Communications on 3G Wireless and Beyond For Computer Communications 26 (17) (2003) 1907–1917.

[16] H. Çam, K. Vadde, Performance analysis of non-blocking OVSF codes in WCDMA, in Proceedings of the International Conference on Wireless Networks, June, 2002 pp. 50–55.

[17] X. Zeng, R. Bagrodia, M. Gerla, GloMoSim: a library for parallel simulation of large-scale wireless networks, Proceedings of the 12th Workshop on Parallel and Distributed Simulations, PADS'98, May, 1998 pp. 154–161.

[18] K.S.J. Pister, J.M. Kahn, B.E. Boser, Smart dust: wireless networks of millimeter-scale sensor nodes, Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Network, 1999 pp. 271–278.

[19] J. Daemen, V. Rijmen, AES proposal: Rijndael http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf, 1999.

[20] R.L. Rivest, The RC5 encryption algorithm, Proceedings of the 1994 Leuven Workshop on Fast Software Encryption, 1995 pp. 86–96.

[21] National Bureau of Standards (NBS), Specification for the data encryption standard, Federal Information Processing Standards (FIPS) Publication 46, 1977.

[22] B. Schneier, D. Whiting, Fast software encryption: designing encryption algorithms for optimal software speed on the Intel Pentium processor, Fast Software Encryption, Fourth International Workshop Proceedings, Springer, Berlin, 1997. pp. 242–259.

[23] B. Schneier, Applied Cryptography, third ed., Wiley, New York, 1996.

[24] A. Sinha, A. Chandrakasan, JouleTrack: A web based tool for software energy profiling, Annual ACM IEEE Design Automation Conference, Proceedings of the 38th Conference on Design Automation, 2001 pp. 220–225.