SPDA: A Security Protocol for Data Aggregation in Large-scale Wireless Sensor Networks

Jin Wook Lee[1] Yann-Hang Lee[2] and Hasan Cam[2]

[1] Networking Lab.,
Samsung Advanced Institute of Technology, P.O. 111, Suwon, Korea 440-600
[2] Department of Computer Science and Engineering,
Arizona State University, Tempe, AZ 85287-8809, USA

Abstract. In this paper we propose a new key establishment protocol enabling any data aggregation protocol to be operated securely. This is accomplished by a bidirectional key distribution scheme based on *Forward Key Setup* and *Backward Key Setup* developed by using synchronized broadcast and multi-level key concept. Our protocol, called SPDA(Security Protocol for Data Aggregation) is well suited for any data aggregation algorithms and applications. Our analysis results prove SPDA's efficiency meaning that its communication cost is manageable.

1 Introduction

The issue of security in sensor networks has been addressed at various levels. In order to prevent unauthorized entities from intercepting, decrypting, or hijacking data communications, data should be encrypted with either symmetric or asymmetric keys. The keys must be protected and managed appropriately between the base station and all sensor nodes and must satisfy several security and functional requirements. To support secured data aggregation in sensor networks, there must be a key management scheme between each sensor node and its correspondent data aggregation node. Thus, the collected data are encrypted at each sensor node and then decrypted at the data aggregation node for aggregation processing. This paper focuses on a simple key establishment for data aggregation in sensor networks. The objective is to construct an efficient key management for data aggregation mechanism that can give confidentiality and integrity against malicious intruders. To eliminate the security loophole open to malicious intruders, we present a mechanism to set up pair-wise symmetric keys for data aggregation operations. The core of a Security Protocol for Data Aggregation (SPDA) mechanism is the ***bidirectional key setup scheme*** that stochastically makes a unique symmetric key between a sensor node and a correspondent aggregation node in the sensor network.

2 Related Works

So far the issue of security in sensor networks has been addressed at various levels. Secure data aggregation schemes have been introduced in recent literatures [1] [2] [3] [4] [5]. In the papers [3] [4], the authors attempted to make data aggregation secure by designing alternate data aggregation schemes such as pattern-based data aggregation and

reference-based data aggregation. They employed a kind of group key management scheme, so a group header becomes an aggregator to perform the aggregation algorithm. However, such schemes also have their limitations in performing specific data aggregation algorithms and specific applications. In another work [2], Przydatek et al. recently proposed secure information aggregation. Their scheme is also dedicated to specific applications. Lee et al. [1] proposed a key management for data aggregation and pointed out that the disadvantage of their scheme was that more data needed to be passed through the key setup message and stored at each node. Worse yet, compromising a node allows an adversary to understand all data communications originating from the node's neighboring nodes. A similar paper to ours is proposed by Hu et al. [5], but they don't discuss key management issues in detail. Unlike other works, we provide a security platform that conforms to any data aggregation scheme and any application scenarios.

3 SPDA PROTOCOL

Our aim is to construct a key establishment mechanism which enable a node to have a unique symmetric key agreed with the correspondent aggregator. Our idea is to allow each node to compute a key with seeds originating from both the Forward Key and Backward Key sent by aggregators. Our design goal is to devise a simple key establishment scheme well suited to any data aggregation algorithm by using pre-defined aggregators. Only aggregators have a data aggregation algorithm to perform and have more security information for key generation. Throughout the paper, we use several terms to describe the protocol as defined in Table 1.

Table 1. Glossary

Glossary	***Description***
FKeySetup	A protocol packet. Only the base station can generate the initial FKeySetup packet. All the nodes start the operation by receiving a FKeySetup packet firstly.
BKeySetup	A protocol packet. Only the aggregators can generate their own BKeySetup packets. All the sensor nodes create their own BKeySetup packet to send data messages to aggregators.
FLK	Forward Level Key. A key hint of the base station.
BLK	Backward Level Key. A key hint of the aggregator.
PubK	Public key of the base station. This public key is shared by all nodes.
SPK	Secret key, partially selected bits of PubK.
PrvK	Private key that the base station and aggregators share.
CK	Combination key. This key is generated by forward and backward key setups and is used by aggregators and sensor nodes.
hc	Hop Count. Logical hop count from the base station
bhc	Backward Hop Count. Logical hop count from an aggregator

3.1 Definition of Keys and functions

Our approach to key generating functions is motivated by the need to establish a symmetric key of each node as efficient in communication cost as possible. Our idea of key generation introduces symmetry of key with two asymmetric keys. Combining two different keys produces a high probability of having a virtually unique key in the network if each of the two different keys is not carelessly generated. One key is propagated from the base station to sensor nodes. The other key is propagated from aggregators to the base station. The keys are propagated by the relaying procedure of each node. The relayed keys should not be easily guessed, so we suggest the use of a one-way hash function as a key relaying function. Each sensor node receives two different keys from two different neighboring nodes by relaying keys in opposite directions.

We define and use three types of crypto keys and two types of seed keys as below:

- Crypto key
  - Public Key (PubK)
  - Private Key (PrvK) of the base station and aggregator
  - Sub-Public Key (SPK)
  - Combination Key (CK)
- Seed key
  - Forward Level Key (FLK)
  - Backward Level Key (BLK)

All nodes maintain the public key (PubK) of the base station. This public key offers data confidentiality of a broadcast message during announcement at the base station. SPK is partially selected bits of PubK (i.e. most significant 64 bits of PubK) and is used as a secret key among all nodes. The public key processing is costly; sensor node's public key processing is performed only during key establishment. Once a sensor node finishes the key establishment, PubK and SPK are no longer used for data confidentiality so this public key mechanism does not significantly affect the network performance. We choose 64 bits for a symmetric key and 512 bits for a public key, so the size of all symmetric keys in this work is 64 bits. Combination Key (CK) is a core cryptographic key in this protocol; it is computed in each node to use for sensed data message confidentiality. Each CK for each sensor node has a high probability of being unique in a network. Additionally, aggregators have the private key (PrvK) of the base station that is pre-installed, so aggregators are said to have the same security power as the base station and can also use this to establish secure channels to other aggregators. During the key establishment, performing Forward and Backward Key Setup enables each sensor node to compute its CK by itself and to use it for data encryption afterwards. Forward Level Key (FLK) and Backward Level Key (BLK) are combined to generate the CK. The first FLK is created only by the base station, whereas the generating BLK is started by all aggregators. We are going to explain how to generate and propagate FLK and BLK in the next subsection.

We suggest applying two one-way hash functions and a combining function to perform the protocol. All nodes have two key-generating functions and one key-combining function as below:

- One-way function

- Forward Key Generating Function (FFunc)
- Backward Key Generating Function (BFunc)

– Combining function
  - Combination Function (CFunc)

3.2 Key Establishment

Our key establishment is done in two stages, *Forward Key Setup and Backward Key Setup*. The base station starts a key establishment phase by broadcasting a message enclosing *The Seed of Forward Level Key*(FLK^l), where l is the level of the base station. We now address the two key setup stages in greater detail, using Figure 1 to help explain the Forward and Backward Key Setup protocol.

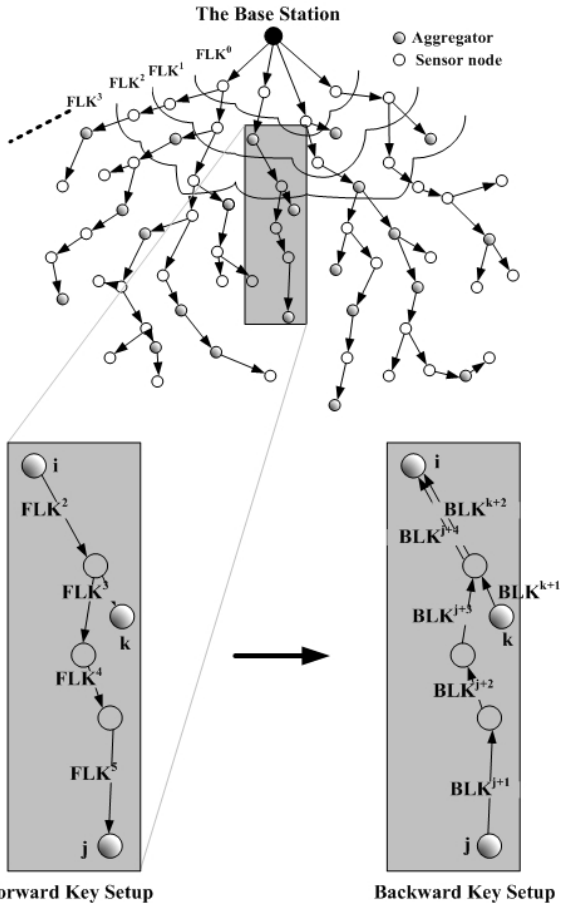


Fig. 1. Network Illustration of Forward and Backward Key Setup

Forward Key Setup Key distribution for wireless sensor networks should not ignore scalability. In order to achieve scalability, a flooding-based broadcast is commonly used for distributing keys. The base station starts the Forward Key Setup stage by sending a $FKeySetup$ packet containing the commitment code of the *Forward Level Key* (FLK^n, where n is a big enough number and the last element of the key chain. For notation, we use superscript for hop count or level) to all adjacent nodes, which prevents an adversary from compromising the base station. In the Forwarding Key Setup stage, the concept of '$level$' is important, meaning that those who have the same hop count from the base station locate in the same level and have the same FLK. The format of the first FKeySetup packet is shown below:

Base Station $\xrightarrow{broadcast}$ ***Neighbors*** : $FKeySetup^0$

$$E_{PrvK}\{\ FLK^n \mid n \mid G\ \}$$
$$\mid E_{SPK}\{\ FLK^0 \mid hc^0\ \}$$
$$\mid MAC_{SPK}(\ FLK^n \mid n \mid G\ )$$

, where G is a gap value of hop-count assigned by the base station.

On receiving the FKeySetup packet, a node starts performing the protocol. The node naively decrypts the packet with PubK and computes the MAC of the decrypted content with SPK. The node then verifies the packet with two procedures, MAC comparison and Commitment key validation. Firstly, the integrity and authenticity of the packet could be validated by comparison of the computed MAC with the received MAC; however, MAC comparison, with the syntax verification method is not enough, since a malicious node that steals SPK is able to forge the whole packet by changing random bits of the content and computing corresponding MAC with SPK. In such a case, MAC comparison could not detect this abnormality. Commitment key validation solves the problem. Provided that *hc* starts with 0 and $increases\ by\ one$, the node applies FFunc with FLK, '$n - G \cdot hc$' times to verify FLK^n. Applying a gap value, G, between two consecutive *hc* prevents an attacker having no G from generating the next FLK properly.

If the verification fails, the node stops the protocol; otherwise, the node becomes a level-1 node and prepares its own FKeySetup packet based on application of FFunc for the next Forward Level Key ($FLK^1 = FFunc^G(FLK^0)$, meaning G times FFunc application). After an appropriate time, all level-1 nodes transmit their FKeySetup packet as below. Nodes that receive these kinds of packets for the first time become a part of the network as level-2 nodes.

Level-1 nodes $\xrightarrow{broadcast}$ ***Neighbors*** : $FKeySetup^1$

$$\text{RELAY}[E_{PrvK}\{\ FLK^n \mid n \mid G\ \}]$$
$$\mid E_{SPK}\{\ FLK^1 \mid hc^1\ \}$$
$$\mid \text{RELAY}[MAC_{SPK}(\ FLK^n \mid n \mid G\ )]$$

, where hc^1 is $hc^0 + 1$.

As a result of completion of the Forward Key Setup stage, all nodes including aggregators have FLK^l, where l is a relative hop distance from the base station.

Backward Key Setup Aggregators may start the Backward Key Setup stage right after they receive a FKeySetup while sensor nodes wait to receive a Backward Key

Setup packet to start the Backward Key Setup stage. Backward Key Setup is performed with unicast communication started by aggregators. Once each aggregator receives a FKeySetup packet, it is ready to start the Backward Key Setup stage by generating the BKeySetup packet as follows (For notation, we use subscript for node ID):

Aggregator i $\xrightarrow{unicast}$ ***Its parent node j*** : $BKeySetup_i$

$$E_{PubK}\{\ Seed_i \mid bhc_i^0 \mid i \mid hc_i\ \}$$
$$\mid E_{SPK}\{\ BLK_i^0 \mid bhc_i^0\ \}$$
$$\mid MAC_{SPK}(\ Seed_i \mid bhc_i^0 \mid i \mid hc_i\ )$$

An aggregator, i, chooses a random number as the Seed of the Backward Key, then applies a one-way function once to make its BLK_i^0. We suggest choosing an arbitrary number for bhc instead of zero. This technique prevents an adversary who compromises SPK from knowing the relative location of an aggregator by calculating bhc. The $Seed$ and bhc are encrypted with PubK and reported to an upper-level aggregator which will use them to compute proper CKs of intermediate sensor nodes in the routing path between two aggregators. Assume that each node chooses one of its neighboring nodes as the next hop node to reach the base station. An aggregator unicasts its BKeySetup to its next hop node (say, node j).

On receiving this BKeySetup packet, sensor node j decrypts $E_{SPK}\{BLK_i^0 \mid bhc_i^0\ \}$ with SPK and makes its own $BKeySetup_j$, which contains updated $bhc_j^1 (= bhc_i^0 + 1)$ and BLK_j^1 (the output of $BFunc(BLK_i^0)$) as below:

Sensor Node j $\xrightarrow{unicast}$ ***Its parent node k*** : $BKeySetup_j$

$$\text{RELAY}[E_{PubK}\{\ Seed_i \mid bhc_i^0 \mid i \mid hc_i\ \}]$$
$$\mid E_{SPK}\ \{\ BLK_j^1 \mid bhc_j^1\ \}$$
$$\mid \text{RELAY}[MAC_{SPK}(\ Seed_i \mid bhc_i^0 \mid i \mid hc_i\ )]$$

Only aggregators having PrvK are capable of understanding all the contents of the BKeySetup. Whenever an aggregator receives a BKeySetup packet, it decrypts the whole packet with PrvK and then keeps the source ID, bhc_i^0, and $Seed_i$ of the source aggregator in the aggregator list.

3.3 Combination Key Generating

The main purpose of Forward and Backward Key Setup is to allow a node to be able to have a secret key agreed with an aggregator for data message confidentiality, which is *Combination Key (CK)*. Once a node generates CK, it could encode a data message to give to an aggregator lightly and securely. Now we explain how each kind of node computes its own CK. There are two sorts of nodes in the protocol, such as aggregator and sensor nodes. Each kind of node does apply differently CK generation functions. A sensor node which receives FLK and BLK uses them as inputs of CFunc as $CK_j = CFunc(FLK_j, BLK_j)$. This key combination is quite unique in the network and also can be computed with seeds of each key. All aggregators share PrvK and are able to compute other aggregator's CK generated with PrvK and their ID as below for aggregator i: $CK_i = CFunc(PrvK, i)$. The reason that an aggregator incorporates its ID into CK is to differentiate CK of other aggregators. Having ID of other aggregators grants an aggregator the CK generation of others.

3.4 Key Usage

Two different kinds of nodes build the first data message with three different message fields usage, as below:

Node k $\xrightarrow{unicast}$ ***Aggregator i*** :

$$E_{CK_k}\{Message\} \mid E_{SPK}\ \{k|0|k\} \mid MAC_{CK_k}(Message) \text{ for aggregators}$$
$$\text{or } E_{CK_k}\{Message\} \mid E_{SPK}\ \{k|bhc_k|aggrID_k\ \} \mid$$
$$MAC_{CK_k}(Message|bhc_k|aggrID_k\ ) \text{ for sensor nodes}$$

, where $aggrID_k$ is the source aggregator ID of bhc that node k receives.

When an aggregator receives the first data message from a node, it searches the source ID, say k, in the aggregator list. If k is found in the aggregator list, the ID is used to compute CK_k. The aggregator figures out the generation of proper CK for sensor nodes.

With aggrID, an aggregator can calculate the length of the path between two aggregators and also locate the message source node with bhc. Therefore, the aggregator is able to find the relative distance of the message source node from aggregators and calculate its FLK and BLK, as illustrated in Figure 2.

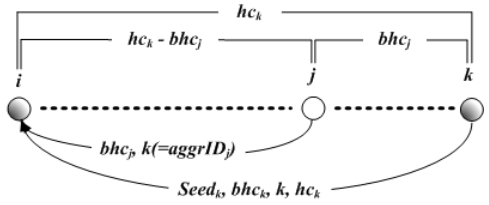


Fig. 2. Finding relative distance of a source node.

As mentioned above, CK calculation is done only once. After that, all nodes send smaller sized messages as below:

Node k $\xrightarrow{unicast}$ ***Aggregator*** :

$$E_{CK_k}\ \{\ Message\ \} \mid k \mid MAC_{CK_k}(Message)$$

4 ANALYSIS

In this section, we show the performance of SPDA through approximate numerical analysis and provide simulation results to validate our numerical analysis. We choose communication cost as the performance metric because the number of communications is critical in wireless sensor network. Communication cost is defined as the additional number of communications per non-aggregator for Backward key setup. We first derive

the basic calculation of the number of nodes. During the Forward key setup, the tree-structured network is formed level by level. The percentage of aggregators (denoted by α) on each level is expected to be the same under uniform distribution. Let $N(l)$ denote the number of nodes on level l. Thus, the number of aggregators, $N_{agg}(l)$, is $\alpha \cdot N(l)$. With a maximum level number of a network, h, the total number of nodes in a whole network, T_N, and the total number of aggregators, T_{agg}, can be calculated respectively as:

$$T_N = \sum_{i=1}^{h} N(i), T_{agg} = \alpha \cdot \sum_{i=1}^{h} N(i).$$

Now we calculate the probability of connections between aggregators. To find the number of non-aggregators between two aggregators along a routing path, we note that they are all located on different levels. In other words, all routing connections between two nodes are between two levels. For example, the percentage of aggregators on a level that have aggregators as their parent nodes is α times the number of aggregators on the level ($\alpha \cdot N_{agg}(l)$ where l is the level number).

Not all nodes receive a BLK necessary to generate CK. In the tree-structured network we target, some nodes are not destined to receive BLK because they are not located in the routing path of any aggregators. We define a *tail node* as a node that has no chance of receiving any BLK from any aggregators and does not contribute the network security. The number of tail nodes contributes directly to the protocol performance as a whole so the number of tail nodes should be found. The number of tail nodes can be derived like this. Intuitively, all non-aggregators on the maximum level l are tail nodes because there is no possibility of getting them to receive a Backward key. In the same way, some non-aggregators on level $l-1$ are going to be tail nodes since they are not selected as parents by aggregators on level l. In the same way, the number of tail nodes on a level with h, maximum level number, can be derived as:

$$N_{tail}(l) = (1-\alpha)^{h-l}(1-\alpha)N(l).$$

The total number of tail nodes in network is

$$T_{tail} = \sum_{i=1}^{h} N_{tail}(i) = \sum_{i=1}^{h} (1-\alpha)^{h-i}(1-\alpha)N(i).$$

We suggest three scenarios for aggregator's BLK forwarding to reduce the number of tail nodes in a network. We categorize three different scenarios for the behavior of an aggregator: (1) An aggregator unicasts a BLK to only its parent node(Say, Scenario I). (2) An aggregator unicasts a BLK to all its parent level nodes, not to just the parent node(Scenario II). (3) An aggregator unicasts a BLK to all its neighbor nodes except its child level nodes(Scenario III).

In an ideal case, just one Backward key is sufficient for the computation of a non-aggregator's CK. However, a node may be expected to relaying several Backward keys to the upper level in SPDA. Note that Backward key setup communication commences from an aggregator and ends at another aggregator. In order to count the number of

additional communications for Backward key distribution, we first find the sum of all intermediate non-aggregators that relay the Backward key for aggregators. t_n represents the number of aggregators in the network that travel through n, the number of non-aggregators, to reach another aggregator. t_n can be derived with α and N_{agg} as below.

$$
\begin{aligned}
t_0 &= \sum_{l=2}^{h} \alpha N_{agg}(l) + N_{agg}(1) \\
t_1 &= \sum_{l=3}^{h} (1-\alpha)\alpha N_{agg}(l) + (1-\alpha) N_{agg}(2) \\
\ldots &= \ldots \\
t_i &= \sum_{l=i+2}^{h} (1-\alpha)^i \alpha N_{agg}(l) + (1-\alpha)^i N_{agg}(i+1)
\end{aligned}
$$

$(1-\alpha)^2 \alpha N_{agg}(4)$, for instance, represents the case when that many aggregators on level 4 send a Backward key; there are 2 non-aggregators relaying the key until the key reaches an aggregator. Therefore, the total number of additional communications necessary for Backward key setup of all non-aggregators is calculated thus:

$$
T_{forward} = p \cdot \sum_{i=0}^{h-1} (i \cdot t_i) = p \cdot \sum_{i=0}^{h} (i \cdot (\sum_{l=i+2}^{h-1} ((1-\alpha)^i \alpha^2 N(l)) + \alpha (1-\alpha)^i N(i+1)).
$$

, where p is the scenario factor. For scenario I, p is 1. p is going to be $0.3N$ and $0.7N$ for scenario II and scenario III, respectively. N is the average number of neighbors. $0.3N$ is the approximate average number of parent level nodes per an aggregator and $0.7N$ is the approximate average number of parent and sibling level nodes per an aggregator.

In conclusion, the average number of forwarding communication per each non-aggregator, $A_{forward}$, is

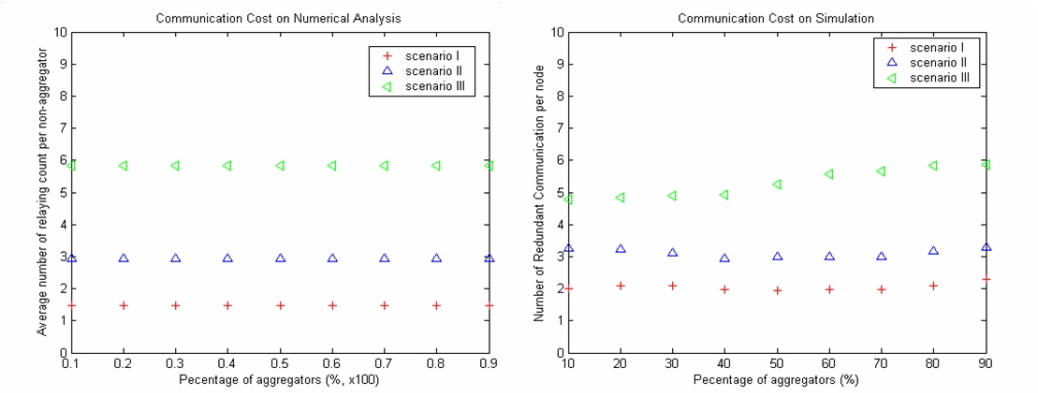
$$
A_{forward} = \frac{T_{forward}}{T_{non}}.
$$

, where T_{non} is $T_N - T_{agg} - T_{tail}$.

Now we can calculate the communication cost if we know the network parameters, such as the average number of neighbor nodes(N), the average number of sensor nodes in a level($N(l)$), the number of aggregators(α), and the maximum number of levels(h). The results of numerical analysis and simulation are shown in Figure 3(a) and 3(b). We set the average number of neighbors, N with 10 and maximum number of level in a network, h with 14 for numerical analysis. We see the numerical results is similar to the simulation results.

5 CONCLUSION

In this paper we proposed SPDA, a security protocol for data aggregation, to offer a lightweight key establishment and adaptability of any aggregation algorithms for large-scale tree-structured sensor networks. The protocol provides data integrity and confidentiality by applying a secret key mechanism established with a key generation and



(a) Result from Numerical Analysis (b) Result from Simulation

Fig. 3. Communication Cost

distribution scheme. The concept of Forward and Backward Key establishment that plays a key role is presented. By using the protocol, a network administrator is able to build secure networks for data aggregation. In SPDA, a sensor node generates its own crypto-key, CK, realizing its uniqueness in a network with a high probability.

There are two main points concerning the proposed protocol. First, SPDA is totally 'distributed'; i.e. there is no central manager for the protocol, such as a cluster-head or group-head. Hence, it is advantageous for scalability and also adding or deleting a sensor node is easy. Second, SPDA is independent of a data aggregation algorithm. Any data aggregation algorithm could be used together with this protocol. Aggregators should be pre-defined before deployment, at the expense of aggregation efficiency., however.

References

1. Y. H. Lee, A. Deshmukh, V. Phadke and J. W. Lee, *Key Management in Wireless Sensor Networks*, Proceedings of the first European Workshop, ESAS 2004.
2. B. Przydatek, D. Song and A. Perrig, *SIA: Secure Inforamtion Aggregation in Sensor Networks*, Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys 2003), 2003.
3. H. O. Sanli, S. Ozdemir and H. Cam, *SRDA: Secure Reference-Based Data Aggregation Protocol for Wireless Sensor Networks*, Proceedings of IEEE VTC Fall 2004 Conference, Sept. 26-29, 2004.
4. H. Cam, S. Ozdemir, P. Nair, D. Muthuavinashiappan and H. O. Sanli, *Energy-Efficient Secure Pattern Based Data Aggregation for Wireless Sensor Networks*, To appear in a special issue of Computer Communications on Sensor Networks.
5. L. Hu and D. Evans, *Secure Aggregation for Wireless Networks*, Proceeding of Workshop on Security and Assurance in Ad hoc Networks, Jan 28, 2003.