

THIS: THreshold security for Information aggregation in Sensor networks

Hai Vu Neeraj Mittal S. Venkatesan
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083, USA

hai.vu@student.utdallas.edu neerajm@utdallas.edu venky@utdallas.edu

Abstract

Message aggregation can be used to reduce the communication overhead in a network where many messages have the “same” content. However, it may make the network less secure because, once the aggregator is compromised, the data correctness cannot be ensured. In this paper, we present a new scheme THIS that allows a base station to accept the aggregated data only if a majority of sensor nodes in the sensing area report “similar” data. This solution enables messages to be aggregated in a secure and efficient manner.

1. Introduction

Sensor networks provide viable solutions for many monitoring problems such as military surveillance and environment monitoring. In many cases, the network is deployed as many-to-one flows from the sensing area to the base station. If the network is such that many packets have the same (or very closely related) content, then the communication overhead for the network can be reduced by combining these packets into a single packet. This merging or *aggregation* of packets can be done by some intermediate nodes on the path between the sensing area and the base station.

Message aggregation can reduce the communication overhead for the network significantly. However, it gives also rise to some non-trivial security issues that need to be addressed, especially if the network is to be deployed in a hostile environment. First, the intermediate nodes, once compromised, can modify, forge, or simply generate false aggregation data. The compromised nodes, therefore, may be able to collectively alter the final aggregation data of the whole network. Second, in order to enable the intermediate nodes to *understand* and aggregate data, message encryption is not feasible if the intermediate nodes do not share a key with the sensor nodes that generated the data. One simple solution for this problem is for all sensor nodes across

the network to share a single master key. However, using the same key for encryption or authentication is undesirable because the adversary can potentially control the entire network by successfully compromising only a single sensor node. Third, an issue related to the correctness of a message aggregation solution is that under what conditions the aggregated data is said to be *correct*, and what aggregation functions are considered secure against a *stealthy attack* [6]. A stealthy attack is defined as one in which the attacker’s goal is to make the base station accept false aggregation results that are quite different from the correct results.

We illustrate the stealthy attack using an example as follows. Consider a battle field with a control network that regulates the enemy’s military force by measuring number of soldiers in each area. It computes the total number of enemy soldiers, and decides whether to raise an alarm for the headquarter. Now, assume that the enemy wants the network to raise a false alarm. It simply captures one of the sensor nodes, generates a report that shown the presence of a large number of soldiers in the battlefield, which is then sent via the aggregated data to the base station. By increasing the sensor’s reading at some areas by hundreds or even thousands, it is able to increase the total number of soldiers in the network, thereby causing a false alarm. This attack is successful because a change in a single reading may lead to a noticeable change to the final summation value. Thus an attacker can succeed in changing the final aggregation result just by fooling a single sensor. The situation becomes more serious if the adversary is able to compromise more sensor nodes and use them collectively to launch a more severe attack.

We propose a THIS (**TH**reshold security for **I**nformation aggregation in **S**ensor networks), which is a scheme to solve the problem that has the following properties:

- *Safety*: it ensures that only the data generated by a majority of sensors at the sensing area should be accepted. Specifically, the data that has been collectively generated or forged by up to t compromised sensors will be

rejected. Our scheme is t -tolerant, this means we allow up to t compromised nodes in the network. Moreover, the scheme ensures that whenever the data is accepted, the reported aggregate values should be “close” to the true values with high probability.

- *Security*: the reports transmitted in the network can be encrypted to protect sensitive information while still allowing aggregation at intermediate nodes at different levels.
- *Efficiency*: symmetric cryptography is used and each sensor only needs to store a few encryption keys.

In Section 2, we discuss why the problem of securely aggregating of information is interesting. The details of the scheme are introduced in Section 3. Some other related work to our scheme are described in Section 4. We conclude the paper in Section 5.

2. Secure Information Aggregation Problem

To see why making message aggregation secure as well as efficient is non-trivial, let us consider the network shown in Figure 1.

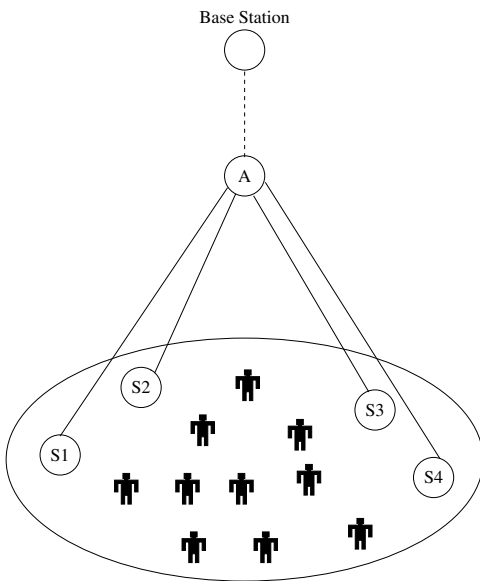


Figure 1. Data aggregation in the sensing area

The sensing area consists of four sensor nodes, $S1$, $S2$, $S3$ and $S4$. Sensor node A is the aggregator for the sensing area and it is also the parent of all the four sensor nodes on the routing path toward the base station. Assume that no sensor node has been compromised. Sensor nodes

in the sensing area collect data and send them to A . Suppose each node reports the same value of 15 soldiers to A . A then computes the aggregation function on all the values (suppose it is the mean function), which returns 15. After that, A transmits the final value to the base station. We can clearly see that by using this simple scheme, the message overhead is reduced by a factor of 4.

Now we consider another scenario in which $S1$ has been compromised. Instead of sending a value of 15 to A , $S1$ replaces it with 75. The mean value computed by A is now 30, which is twice as much as the real value. We may argue that A can detect the big difference between the value sent by $S1$ and that of other sensor nodes, hence rejecting that possibly wrong value. Then in that case the adversary may compromise more sensor nodes, for instance $S2$, so that the difference in the reading of each sensor cannot be detected easily. Specifically, should A discard the reading from $S1$ and $S2$, or from $S3$ and $S4$?

In the worst case, the aggregator itself is compromised. Because the sensor nodes which sense and report the data may not be able to know about the final aggregated value sent by the aggregator, the aggregator can modify or generate the data arbitrarily. In that case the base station will find it very hard to verify the correctness of the data. In our example above, assume all the sensor nodes $S1$, $S2$, $S3$ and $S4$ are good and report the real data (which is 15), A is a compromised node and sends the wrong modified data of 75 to the base station. When receiving the report, the base station may not be able to easily verify if this value is correct.

In the following sections, we describe our solution in detail and show that it solves all the issues that we describe in this section.

3. THIS: A Secure Information Aggregation Scheme

3.1. Assumptions

Before describing the idea of our scheme, we make some assumptions as follows:

- We assume that each sensor node shares a symmetric encryption key with the base station, which is unknown to other sensors in the network. In addition, we assume that a sensor can securely set up pair-wise keys with each of its neighbor nodes.
- We assume that the network is dense enough so that each sensor area consists of at least $t + 1$ sensor nodes which will collect the data and send the report to the base station. t is a threshold parameter represents the number of compromised nodes that our scheme allows

in the network, i.e., we assume there are no more than t compromised nodes in the network. The compromised nodes may be located anywhere in the network, including in the same sensing area.

- The sensor nodes in the same sensing area are able to form a cluster with a node being selected as the cluster head. We can imagine that the cluster is organized like a tree with the cluster head as the parent and all other nodes as the leaves of the tree. However, the technique to be used to construct and maintain the cluster does not concern us.

3.2. Description of the scheme

Our THIS protocol consists of three phases: the *setup* phase, the *aggregation and certification phase* phase and the *authentication* phase. The details of each phase are as follows:

- *Phase 1 (setup phase)*: We divide this phase into four steps:
 - *Step 1 (selecting secret keys)*: Before deployment, each sensor node is configured with one or more symmetric keys selected from the key pool of the base station. Note that each sensor node shares some secret keys with the base station, but different sensor nodes do not know the keys of each other.
 - *Step 2 (setting up the cluster)*: After deployment, all sensor nodes in the sensing area set up the cluster and select a cluster head. Note that the cluster head can also sense the data like a normal sensor in the cluster. The cluster head acts as the aggregator for the cluster. From now on, we use the two terms “cluster head” and “aggregator” interchangeably. The cluster head should be selected such that it has a routing path to the base station. The cluster head may need to spend more energy for collecting the data from other sensor nodes, performing the aggregation function and transmitting the final data to the base station. Because of this, the role of being the cluster head might be changed among the sensor nodes in the same cluster. At the end of this step, the nodes of the cluster are organized as a sub-tree in which the root is the cluster head and the leaves are all other sensor nodes in the cluster. The only requirement of the cluster is that it must consist of at least $t + 1$ sensor nodes.
 - *Step 3 (setting up pair-wise keys)*: In this step each sensor node sets up the pair-wise key with

its cluster head. In [9] and [10], the authors describe methods to securely set-up the pairwise key between any pair of nodes in the network. We use this method as a building block for our solution. After this step, we assume that each node in the network (including the cluster head) shares a secret key with its parent on the routing path toward the base station.

- *Step 4 (setting up cluster keys)*: Once the cluster is formed and the cluster head has a shared pairwise key with each of its children, it then randomly generates a cluster key and sends this key to all of the nodes in the cluster using the pairwise key that was set up in step 3. In order to increase the security, periodically the aggregator may select a new cluster key and propagate the new key to the cluster nodes.
- *Phase 2 (aggregation and certification phase)*: We also divide this phase into four steps.
 - *Step 1: (sensing data)*: The leaf nodes in the cluster (and maybe the aggregator itself) sense the data and transmit the data to the parent, which is the cluster head. Note that all the transmitted data from the leaf nodes to the cluster head are encrypted using the pair-wise keys.
 - *Step 2: (aggregating data)*: The aggregator computes the aggregation value based on the values it has received from the cluster nodes. Once the computation is done, the aggregator encrypts the final aggregation value using the cluster key and broadcasts this as the final value to all the nodes in the cluster. We denote the final value as $finalV$.
 - *Step 3 (certifying aggregated data)*: The sensor nodes that receive $finalV$ check if this value is “close” to the value that they have proposed in step 1. We will discuss later about what it means for one piece of data to be “close” to another. If a sensor i “agrees” with the value sent by the aggregator, it generates a signature $s_i = MAC(key_i^{BS}, finalV)$ where key_i^{BS} is the secret key that i shares with the base station and MAC is the Message Authentication Code. Then i sends this signature back to the aggregator. If, on the other hand, a sensor node finds that this value is not consistent with its proposed value, it refuses to generate the MAC for the final value.
 - *Step 4 (reporting certified data)*: After collecting sufficient number of different signatures, the aggregator encrypts the packet, which includes

all the signature s_i and the final value $finalV$ computed previously, and then transmits it to the base station. It also signs this packet using the key shared with base station. This is for the base station to check the integrity of the packet.

- *Phase 3 (authentication phase)*: Assume that the base station succeeds in receiving the packet from a cluster. Since the base station has the keys shared with all the nodes in the network, it can decrypt the packet and check the contents. If the packet contains less than $t+1$ distinguished signatures, then the base station discards the packet immediately. If at least $t+1$ signatures in the packet are consistent with the final value, then the data is accepted. On the other hand, if the base station is unable to find at least $t+1$ consistent signatures, then there must be something wrong with the packet. That is, either the aggregator and/or the leaf nodes are compromised, or the data has been modified en-route. Later in section 3.6 we will analyze all the possibilities that lead to inconsistencies in a packet.

3.3. Pseudocode

We now provide the pseudocode for different types of sensor nodes in our scheme. There are 3 different types of nodes: the cluster's sensor node, the aggregator and the base station. In the pseudocode, we use the following notations:

- CH : the cluster head (or aggregator).
- BS : the base station.
- $k_i^j = k_j^i$: the key shared between i and j .
- k_c : the cluster key.
- r_i : the plain data sensed by the sensor node i .
- c_i : the encrypted data.
- s_i : signature.
- $MAC(k, r)$: compute the message authentication code.
- $E(k, r)$: the encryption function with key k and data r as input.
- $D(k, c)$: the decryption function with key k and encrypted data c as input.
- $finalV$: the final aggregated data.
- t : threshold parameter.
- n : number of sensor nodes in the cluster, $n \geq t + 1$.
- m : number of actual reports, $m \leq n$.
- m' : number of actual signatures, $m' \leq m$.
- $\Delta(r, f)$: the function to check whether the final value is "close" to the reported data.

3.4. Security analysis

We now argue the correctness and the security of our solution. Right now we assume that the network enables

Algorithm 1 BASE STATION

```

1: /* phase 3 */
2: receive  $c_p$  and  $S$  from  $CH$ 
3: decrypt  $p' = D(k_{CH}^{BS}, c_p)$ 
4: compute  $S' = MAC(k_{CH}^{BS}, p)$ 
5: if  $S'$  and  $S$  do not match then
6:   discard packet
7: end if
8: /* check  $p'$  */
9: if there are less than  $t + 1$  signatures then
10:  discard packet
11: end if
12:  $con := 0$  /* number of consistent signatures */
13: for each sensor  $i$  in the identity list do
14:  compute  $s'_i = MAC(k_i^{BS}, finalV)$ 
15:  if  $s'_i$  matches  $s_i$  then
16:     $con := con + 1$ 
17:  end if
18: end for
19: if  $con \geq t + 1$  then
20:  accept packet
21: else
22:  discard packet
23: end if

```

Algorithm 2 AGGREGATOR

```

1: /* step 2 - phase 2 */
2: received  $m$  reports
3: if  $m < t + 1$  then
4:  stop algorithm
5: end if
6: for each  $c_i$  that  $CH$  received do
7:  decrypt  $r_i = D(k_i^{CH}, c_i)$ 
8: end for
9: compute  $finalV$ 
10: encrypt the final value  $c_f = E(k_c, finalV)$ 
11: broadcast  $c_f$  to all sensor nodes
12: /* step 4 - phase 2 */
13: received  $m'$  signatures
14: if  $m' < t + 1$  then
15:  stop algorithm
16: end if
17: create a packet  $p$  composing of  $finalV$ ,  $m'$  signatures
   and their corresponding sensor's identity.
18: sign on the packet  $S = MAC(k_{CH}^{BS}, p)$ 
19: encrypt  $c_p = E(k_{CH}^{BS}, p)$ 
20: send both  $S$  and  $c_p$  to  $BS$ .

```

Algorithm 3 CLUSTER'S SENSOR NODE

```
1: /* step 1 - phase 2 */
2: sense the data, save as  $r_i$ 
3: encrypt  $c_i = E(k_i^{CH}, r_i)$ 
4: send  $c_i$  to  $CH$ 
5: /* step 3 - phase 2 */
6: receive  $c_f$  from  $CH$ 
7: decrypt  $finalV = D(k_c, c_f)$ 
8: if  $\Delta(r_i, finalV) = TRUE$  then
9:   compute  $s_i = MAC(k_i^{BS}, finalV)$ 
10:  send  $s_i$  to  $CH$ 
11: end if
```

only one level of aggregation, later on we will relax this restriction to allow a higher level of aggregation. First, we analyze the case when one of the compromised nodes is the aggregator. Under the stealthy attack, once the aggregator is compromised, it may try to generate false data and make the base station accept this. However, since there are no more than t compromised nodes, the compromised aggregator (together with other $t - 1$ compromised nodes) is able to collect at most t legal signatures. In our scheme, if the base station checks the received packet but could not find at least $t + 1$ legal signatures, it rejects the data immediately. Therefore, in order to fool the base station, the compromised aggregator needs to collect at least $t + 1$ different legal signatures, meaning the adversary needs to capture at least $t + 1$ nodes. However, under our assumption, this is not possible. Hence, compromising the aggregator does not help the adversary to successfully launch the stealthy attack.

We now analyze the security of our scheme when cluster's sensor nodes are compromised. If the compromised nodes propose a value which makes the final aggregation value deviate far away from the acceptable bound, then the other leaf nodes reject this value and do not generate the signature for the final value. For example, consider the same attack that we have discussed in Section 1. The correct value for the cluster should be 15 soldiers. Let the acceptable bound be 5. The value proposed by the compromised nodes which made the final value fall outside of this bound (say, 30 soldiers) is rejected by the correct nodes which proposed 15. In this case, either the (good) aggregator tries to find other sensor nodes or refuses to send out the packet.

Let us get back on when a sensor node can tell if the aggregated final data is "close" to its proposed data. The definition of the function $\Delta(r, finalV)$ depends on the application. For example, a sensor node accepts the final aggregated value only if its proposed value deviates by no more than 10%. Let the acceptable bound be d , then we can simply define the checking function as follows:

$$\Delta(r, finalV) = \begin{cases} TRUE & \text{if } |finalV - r| \leq |r| * d \\ FALSE & \text{otherwise} \end{cases}$$

Applying this function to the above values, we got $|finalV - r| = |30 - 15| = |15| > 15 * 0.1 = 1.5$. Hence $\Delta(15, 30)$ returns *FALSE* and the sensor will refuse to generate the signature.

On the other hand, if all the sensor nodes are good and report the value of 15, then $|finalV - r| = |15 - 15| = 0 < 15 * 0.1 = 1.5$. $\Delta(15, 15)$ returns *TRUE* and the sensor will accept to generate the signature.

In reality, it depends on the property of the application data that we can define the appropriate function Δ .

3.5. Overhead analysis

Now we analyze the communication and computation overhead of our scheme. The first phase of our scheme, which is the setup phase, needs to be performed only once. Specifically, selecting the secret key shared between each sensor and the base station can be done in the pre-distribution stage. Thus there is no communication overhead involved. Setting up the cluster can be done by existing technique and does not require any special clustering method. Therefore, in the *setup* phase, the main communication overhead occurs when we set up the pair-wise key and cluster key between nodes in the cluster and the cluster head. However, this is a one-time operation, because after this phase, the aggregator is able to communicate with all the nodes in the cluster through broadcasting.

In phase two, each sensor node on the field generates the data, sends it to the aggregator and lets the aggregator forward the data to the base station. Assume that the total number of nodes in the cluster is $n \geq t + 1$. In each iteration, each node sends its data to the cluster head once, which costs n messages. Then the cluster head broadcasts the aggregated value, which costs only one message. Eventually, each sensor sends its signature of the aggregated value to the aggregator and this costs another n messages. Now we assume that the routing path from the cluster to the base station consists q hops. Assume that the size of the aggregated packet (including signatures and identities) is larger than the normal packet by 10%. Since only one (aggregated) message needs to be sent, it costs $1.1 * q$ messages for the whole cluster. In each iteration, therefore, we need to spend $n + 1 + n + 1.1 * q = 2n + 1.1 * q + 1$ messages. Without performing the aggregation, it would cost us $n * q + n$ messages for each iteration. Doing some simple calculations, we can easily see that our scheme significantly reduce communication overhead if q is sufficiently large.

Phase three is the checking and authenticating phase and does not require communication overhead.

Considering the computation overhead, we use standard symmetric encryption/decryption algorithms which are suitable for power-limited sensor nodes.

In overall, in each of the phases, our scheme always tries

to reduce the communication and computation overhead as much as possible while still maintaining a high level of security.

3.6. Discussion

Now we discuss the cases in which the base station discards a packet received from a cluster head. Based on our protocol, the following cases may happen:

- The packet has been modified en-route. This case happens when the base station detects the inconsistency between the content of the packet and the signature signed by the cluster head. Since no one knows the secret key shared between the cluster head and the base station, there must be some compromised nodes en-route trying to forge the packet. Our scheme efficiently defense against this attack. We eliminate the case when the adversary has captured the cluster head, because if it is so, then the adversary would not need to modified the packet en-route.
- There are less than $t+1$ signatures in the packet. By the protocol, the cluster head should not send any packet to the base station when it has not collected at least $t + 1$ different signatures. This implies that the cluster head has been compromised.
- There are not enough $t + 1$ consistent signatures in the packet. This case implies that there are some compromised nodes in the cluster.

In any case, by combining the security check at each level: the sensor nodes, the cluster head and the base station, the base station would never accept the false data, hence the network has efficiently been defended against the stealthy attack.

In the network, if the data from each cluster has the same characteristics and is closely related, then it is possible to aggregate data at multiple levels. On the other hand, if data in one cluster is completely independent from another, then aggregating data at a level beyond the cluster-head is not likely to be useful. For example, consider an application where we need to control the overall temperature of a multi story building. If we imagine each room is a cluster, then aggregating data at each floor is reasonable, because data (which is a number) in one cluster is related to another. However, in order to maintain the correctness of our scheme, at each level of aggregation, there must be at least $t + 1$ different packets from the lower level children for the parent to be able to aggregate the data.

4. Related work

Wagner [3] describes attacks on standard schemes for data aggregation and introduces the problem of securing aggregation in the presence of malicious or spoofed data. The robustness of some aggregation functions against malicious data is also quantified. Specifically, Wagner reasons that some functions are inherently insecure (such as average) and discusses/analyzes the aggregation functions which provide better security level. Some of which are *estimation of location, median, count, truncation and trimming*. These functions may help in evaluating and quantifying data aggregation functions in sensor networks. One may carefully select the appropriate aggregation functions and apply our scheme to archive a complete application which is secure internally and externally.

Hu and Evans [4], propose a solution that is resilient to intruder devices and single device key compromises. They assume that the network is organized like a tree, in which only the leaf nodes monitor the area, generate the data and transmit them to the base station at the root of the tree. The intermediate nodes only aggregate and forward the data to the base station. Their approach is based on two ideas: delayed aggregation and delayed authentication. That is, instead of aggregating messages at the immediate next hop to the leaf nodes, the data messages are forwarded unchanged over the first hop and will be aggregated at the second hop toward the base station. The reason to do this, even it increases the communication overhead, is because they assume that the probability of two consecutive nodes to be compromised is low, thus enabling integrity guarantees for the networks. This scheme works fine under this assumption. The protocol cannot effectively prevent a compromised leaf node from generating bogus readings, because its parent just forwards the data to the base station. Moreover, if the adversary can compromise both the children and the parent nodes in the same hierarchy, then a stealthy attack involving these node can be launched and the protocol will not detect this.

In [6], Przydatek et. al. propose a protocol for securely computing of several aggregation functions, such as the median and the average of the measurements, for estimating the network size and for finding minimum and maximum sensor readings. The protocol only requires sub-linear communication between the aggregator and the user. It is also resilient to a malicious aggregator and sensor nodes. They have shown that even some aggregation functions which are inherently insecure can be securely computed in the presence of the compromised aggregator. The authors focus mainly on designing efficient protocols to detect the aggregator cheating if the data is out of a defined bound, but they do not solve the problem when the sensor nodes which generate the reading data have also been compromised.

Mahimkar and Rappaport [7] propose a different approach for solving the aggregation problem. They assume the network is organized as clusters, each cluster has a cluster head with the responsibility to collect and aggregate the data. The authors present a protocol for establishing cluster keys using verifiable secret sharing. Using the cluster key as a basic block, they propose a protocol that ensures that the base station never accepts faulty aggregated readings. This solution depends on several sub-routines, such as elliptic cryptosystem to provide asymmetric encryption, Lagrange interpolation for setting up cluster key and Merkle hash tree.

5. Conclusion

In this paper, we have studied the problem of aggregating data in sensor networks, which is very important to reduce the communication overhead caused by forwarding redundant sensed data.

We present THIS, a new solution for securely aggregating information of sensed data. Our solution uses efficient symmetric cipher and is tolerant to the compromise of up to t nodes including the leaf nodes and the aggregator. The solution can be extended to allow higher levels of aggregation rather than just one level at the local sensing area. THIS is secure in term of defending against stealthy attack, while using the efficient encryption algorithms for reducing overhead.

References

- [1] C. Intanagonwiwat, D. Estrin, R. Govindan and J. Heidemann. "Impact of network density on data aggregation in wireless sensor networks". In *Proceedings of International Conference on Distributed Computing Systems*, Vienna, Austria, 2002.
- [2] B. Krishnamachari, D. Estrin, and S. Wicker. "The impact of data aggregation in wireless sensor networks". In *IEEE International Workshop on Distributed Event-Based Systems (DEBS)*, 2002.
- [3] D. Wagner. "Resilient aggregation in Sensor Networks". In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, Washington DC, USA, 2004.
- [4] L. Hu and D. Evans. "Secure aggregation for wireless networks". In *Workshop on Security and Assurance in Ad hoc Networks*, 2003.
- [5] A. Perrig, R. Szewczyk, V. Wen, D. Culler and D. Tygar. "SPINS: Security Protocols for Sensor Networks". *Wireless Network Journal (WINE)*, 2002.
- [6] B. Przydatek, D. Song, and A. Perrig. "SIA: Secure information aggregation in sensor networks". In *Proceedings of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys 03)*, Los Angeles, USA, 2003.
- [7] A. Mahimkar and T. S. Rappaport. "SecureDAV: a secure data aggregation and verification protocol for sensor networks". In *IEEE Global Telecommunications Conference (GLOBECOM 04)*, 2004.
- [8] E.-O. Blass and M. Zitterbart. "An efficient key establishment scheme for secure aggregating sensor networks". In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security (CCS 06)*, Taipei, Taiwan, 2006.
- [9] W. Du, J. Deng, Y. Han and P. Varshney. "A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks". In *Proceedings of 10th ACM Conference on Computer and Communication Security (CCS 03)*, Washington DC, 2003.
- [10] D. Liu and P. Ning. "Establishing Pairwise Key in Distributed Sensor Networks". In *Proceedings of 10th ACM Conference on Computer and Communication Security (CCS 03)*, Washington DC, 2003.
- [11] E. Heinzelman, A. Chandrakasan and H. Balakrishnan. "Energy Efficient Communication Protocol for Wireless Microsensor Networks". In *Proceedings of 33rd Annual Hawaii International Conference on System Sciences (HICSS)*, 2000.