Implementation of Domingo Ferrer's a New Privacy Homomorphism (DF a New PH) in Securing Wireless Sensor Networks (WSN)

Levent Ertaul
Department of Mathematics & Computer Science
California State University, EastBay
Hayward, CA, USA.
levent.ertaul@csueastbay.edu

Johan H. Yang
Department of Mathematics & Computer Science
California State University, EastBay
Hayward, CA, USA.
s3cur3lab@gmail.com

Abstract- *Wireless Sensor Networks' (WSN) Data aggregation is substantial in eliminating information redundancy and increasing the lifetime of the network. However, in its implementation, the data are being transmitted in clear hence are prone to security attacks. The Homomorphic Encryption Schemes (HES) provide the most advantage in securing the data on limited-resourced WSN devices by allowing operations to be performed on the ciphertext as if the operations are performed on the plaintext. The other conventional encryption schemes fall short compared with HES because they demand too much of resources. The HES that secures WSN's data aggregation can conserve the much needed power source while providing the security. When HES is used, the End-to-End encryption is achieved where the information is secured from the source to the destination. In this paper, we show the performance, the feasibility and the scalability of Domingo Ferrer's a New Privacy Homomorphism (DF a New PH) on large scale simulation framework OMNet++. In this simulation MICA2 sensor nodes are used and up to 11000 sensors are deployed. We also show that DF a New PH algorithm, implemented in C++, has fast performance and low power requirements, feasible to be implemented inside MICA2 sensor nodes and scalable to thousands of sensor nodes without straining the lifetime of the whole network.*

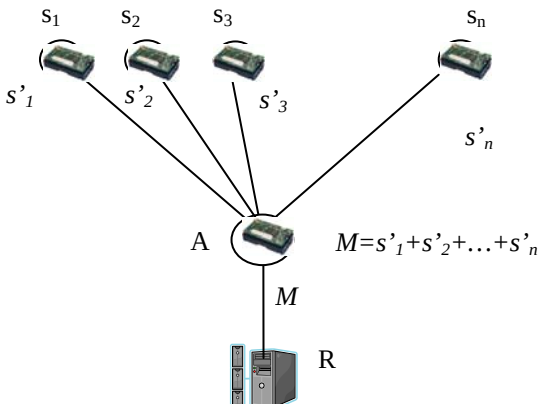
Index Terms—Wireless sensor networks, Data aggregation, Homomorphic encryption schemes.

1 Introduction

A WSN consists of inexpensive and low power sensor nodes that are capable of computation, storage and communication. The sensor node by itself has limited capabilities: *limited computing resource* (slow computing, small storage, and limited power source), *unreliable type of communication* (unreliable data transfer, limited data rate, and limited communication range), *unattended operation* (limited trust, complex remote management). However, adding more nodes, these seemingly limited devices start creating a powerful infrastructure which a user can use for a particular application of providing high resolution information about sensed phenomena[1] [2] [3] [4] .

Because of the nature of WSN data driven networks, it raises new threats that are different from what we faced before. WSN allows massive data collection, coordinated analysis, and automated event correlation. For instance, WSN enables routine tracking of people/vehicles over long periods of time, with troubling implications. With that in mind, the need of securing the data kept inside WSN is paramount [1] [2] [3] [4] [5] [6] [7] [8] [9].

The data gathered by the sensor nodes in most scenarios are environmental data (e.g. temperature) which will be processed inside the network and will end up at one central point, the Sink node. Since the energy consumption increases linearly with the amount of transmitted data, a data aggregation approach helps increase the overall lifetime of WSN [6] [7] [8] [9].



S_1 to S_n are the sensor nodes, A is the aggregator node and R is the sink node. S'_n are the value sensed by the sensors. M is the aggregator function (addition)

Figure 1: Representation of Sensor node, Aggregator node and Sink node in WSN

In WSN, the nodes are functionally separated into sensor, forwarding, aggregator, and sink node. Figure 1 explains the data flow from sensor nodes to sink node. Sensor nodes sense the environment and send the data to the aggregator node. Aggregator node aggregates the data received from the sensor nodes. The aggregation function may be the function of Counting (sum, average, etc), Comparing (maximum, minimum), or Detecting (motion). The forwarding node just forwards the data and its function usually is combined with the aggregator node. Sink node, assumed to have more resource than the other nodes, is where the information ends up [6] [7] [8] [9] [10].

Aggregator nodes, located in the middle of the network, are maintaining the infrastructure of the network. The lost of an aggregator node situated on the network critical point might cripple the network. Whereas the sensor nodes, located on the edges of the network, persist mostly in sleep mode until the sink node queries the network or the environment triggers the event which requires them to activate [6] [7] [8] [9].

Two important issues in securing WSN are node capture and power conservation. Wide deployment in an open area makes WSN prone to node attacks. Most conventional security measures are infeasible to be implemented in limited-resource WSN because they consume too much energy which shortens the node lifetime[1] [2] [3] [5] [11]. The HES can secure WSN's data aggregation while conserve the much needed power source [6] [7] [8] [9].

The HES have the property of additive or multiplicative homomorphism. In *additive homomorphism*, decrypting the sum of two ciphertext is same as addition of two plaintext *(x,y)* represented as $x+y=D_k(E_k(x)+E_k(y))$. In *multiplicative homomorphism*, decrypting the product of two ciphertext is same as multiplication of two plaintext. Multiplicative homomorphism is mathematically represented as $x*y=D_k(E_k(x)*E_k(y))$. The advantage of using homomorphic encryption is that the intermediate aggregator node need not decrypt and then encrypt data to perform the aggregation operation [6] [7] [8] [9]. It can perform the aggregation operation using only encrypted data.

Before committing time and money on the actual WSN deployment, we used a robust simulation framework, OMNet++[12] based on C++[13], to test the feasibility of the DF a New PH scheme on the MICA2[14] sensor nodes. Up to 11000 MICA2 sensor nodes are deployed in this simulation.

This paper is organized as follows. In section 2, we briefly describe the overview of homomorphic encryption schemes. In section 3, we briefly describe the implementation of DF a New PH in the simulation. In section 4, we discuss the simulation results. Finally, in section 5 conclusions are given.

2 Homomorphic Encryption Schemes (HES)

In this section, we give an overview of the HES advantages which benefit WSN. And also one particular HES, DF a New PH scheme is discussed.

2.1 The HES Advantages

Formally HES is described as: given two ciphertext resulted from two values *x* and *y*, one can do operations directly on the ciphertext without knowing the plaintext or the decryption keys[5] [6] [7] [8] [9] [15].

Traditional WSN data aggregation requires repeat cycles of *decryption-re-encryption-operation* every time the sensor node sends encrypted data. The HES cuts down those steps. The sensor nodes can spend most of their time in sleep mode which in turn prolong the node lifetime.

End to End Security, where the information is concealed from the source through the destination, is achieved because the infrastructure nodes, such as aggregator nodes, do not have to keep or know any secret keys[6] [7] [8] [9].

2.2 Domingo Ferrer's a New Privacy Homomorphism (DF a New PH)

DF a New PH is introduced in [16] which is a homomorphic encryption scheme not vulnerable to Known Ciphertext Attacks[17]. Once the value is acquired by the sensor node, the algorithm splits the value into several parts. Each of split value then is encrypted and sent to Aggregator node. The values are then can be added in the aggregator node. Only the Sink node will be able to decrypt aggregated data to get the real value.

Let us look into the protocol in detail as given in [6] [8][16]. In this protocol *n* and *m* are the public parameters (available to every sensor node). The secret parameters, two large prime numbers *p* and *q*, where $m = p * q$, are only available to the Sink and sensor nodes. The number '*n*' represents the split of the plaintext. The secret keys are p, q, x_p, x_q. Here, $x_p \in Z_p$ *and* $x_q \in Z_q$.

Encryption operation is performed by selecting the plaintext $a \in Z_m$. We then split *a* into secret numbers $a_1, a_2 \ldots a_n$, such that $a = (a_1+a_2 ..+a_i+..a_n) \bmod m$ and $a_i \in Z_m$. $E_k(a) = (a_1 x_p \bmod p, a_1 x_q \bmod q), (a_2 x^2_p \bmod p, a_2 x^2_q \bmod q) \ldots (a_n x^n_p \bmod p, a_n x^n_q \bmod q)$

Decryption operation is performed by computing scalar product of the i^{th} pair *[mod p, mod q]* by $[x^{-i}_p \bmod$

$p, x^{-i}_q \bmod q]$ to get $[a_i \bmod p, a_i \bmod q]$. The pairs are then added up to get $[a \bmod p, a \bmod q]$. Finally, Chinese remainder theorem (CRT) [18] is performed to get $a \bmod m$.

This homomorphic scheme has the property of additive and multiplicative homomorphism. We have to acknowledge that this scheme is not secure against Known Plaintext Attacks [17].

3 Implementation of DF a New PH algorithm on Large Scale Simulation

OMNet++ [12] [19], based on C++[13], is a discrete event driven platform in which we perform the *steady state* simulation. Even though the *transient state* simulation which focuses on the initial configuration is also important, we focus on the long term effect of the security implementation to the performance of the sensor nodes[20].

The simulator enables us to control the time frame and the level of details of the results. When it performs the encryption, the simulation goes into *unit time advance* which captures the minute details of the CPU, Memory and Radio operations. When the sensor nodes are in the sleeping mode, the simulator goes into *event advance* which progress the simulation quicker [20].

To identify the algorithm parameters which give the best performance and the longer lifetime along with the desired security, we will compare the performance of the sensor node in performing the algorithm on different parameters. On larger scale, we simulate the sensor node within a cluster and within a network, up to 11000 sensor nodes to identify the performance of the aggregator node.

3.1 The measurements of Performance, Feasibility, and Scalability.

In order to compare the Performance, the Feasibility and the Scalability of the implemented encryption scheme, we measure the *power usage* (in *milliAmpere*) and the *execution time* (in μs) that a sensor node requires to complete its operation.

The dynamics of *power usage* and *execution time* are important to determine the node's lifetime and ultimately to the network's lifetime. For example, one activity might consume high energy, but if it is done sparingly, it might still be feasible for WSN. On the other hand, a small power usage activity, like SLEEP, if done continuously might tremendously shortened the node's lifetime[21].

On the simulation, we compare of three major energy consumer node devices; they are: the CPU, the Memory, and the Radio. The CPU will be active the whole time while the sensor node is encrypting the data.

The memory has two states: READ and WRITE states, where the WRITE state consumes higher energy than READ states. The ideal algorithm is to have lesser WRITE states and lesser *execution time*. The Radio module will employ several Transmission power level depending the inter-node distant, the length of the message, and traffic condition of the wireless communication channels [21] [22] [23].

3.2 The Simulation Specifications

The simulation environment is an Intel Pentium Xeon dual-core 2GHz, 2GB RAM at CompCore facility at CSU East Bay (www.compcore.csueastbay.edu). The Operating System is Ubuntu Linux 6.10 [24]. The code is developed in C++[13] .

Due to the in depth research and the wide adoption on current WSN applications, we based the simulation on MICA2™ sensor nodes[14][22].

The randomly sampled sensor node is situated within a cluster of sensor nodes consists of ten sensor nodes and one aggregator node (Figure 2). The sampling size is from 100 simulation runs to get the average performance of the sensor node in performing the same encryption with the same plaintext size and key size parameters.

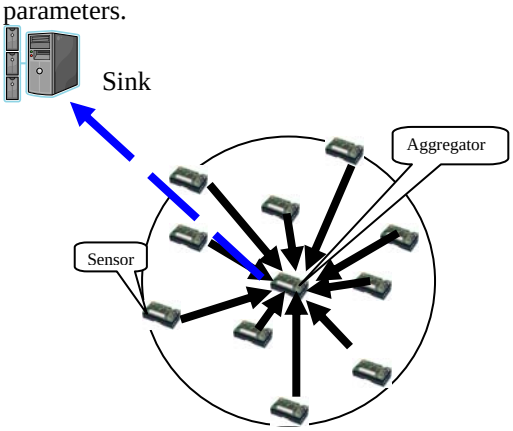


Figure 2: Simulation configuration

Figure 2 shows the simulation configuration of one cluster of sensor nodes. The ten sensor nodes are reporting the sensed data to the Aggregator node in the middle. After the data values are added up (we selected addition aggregation function in this simulation), the result is sent to the Sink. Then the Sink decrypts the aggregated data to recover the real value.

On larger level simulation, the network level, we simulate the cluster of clusters nodes to see the effect of

larger number of sensor nodes, up to 11000 sensor nodes, to the lifetime of the aggregator node.

4 Simulation Results

We measure one cycle of node operation on the sensor node performing the following tasks: *reading the random environment value, performing the encryption on the value, putting the encrypted value into 802.11 MAC* [25][26] *data frame, waiting for clear wireless channel, transmitting the data frame,* and *putting the node into SLEEP* state until the next sampling time interval.

To illustrate the simulation result, Figure 3 is used. OMNet++'s Plove[12] captured the four lines which depict the CPU operation (blue line), the Memory (red line), the Radio (green line), and the orange line which is the *sum* of these three operations (total power usage). In the next discussions, we only use the *sum* line to compare different parameters used in DF a new PH algorithm.

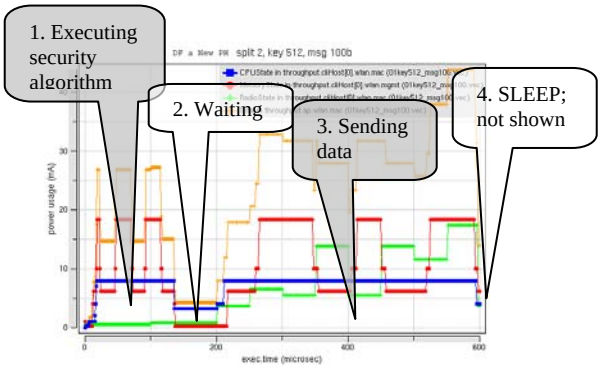


Figure 3: The detailed graph result example

Figure 3 shows one cycle of sensor node operation as an example. It consists of four major activities. It starts with *executing the security algorithm*, *waiting* for clear channel, *sending the data*, and *SLEEP* (not shown on the graph).

On the first activity, *Executing the security algorithm*, it shows the CPU is active from 0 to 140µs. During that time the node completes the following tasks: *getting the random environment value, performing the encryption on the value, putting the encrypted value into 802.11 MAC data frame.* The Memory is also performing READs (the bottom red-lines) and WRITEs (the upper red-lines). The Radio is idle most of the time reflected on the low green line.

The node is *Waiting* for 80µs for clear channel, before it send the data over to the Aggregator node.

On the third activity, *Sending the data*, shows all three devices activated. During the 400µs, the CPU is active the whole time, the Memory performs 3 READs and 3 WRITEs, the Radio goes up and down reflecting the use of several levels of Transmission power [23].

The fourth activity, *SLEEP*, even though it also consumes energy, is omitted from the graph because we only want to focus on comparing the performance of sensor node in executing the algorithm.

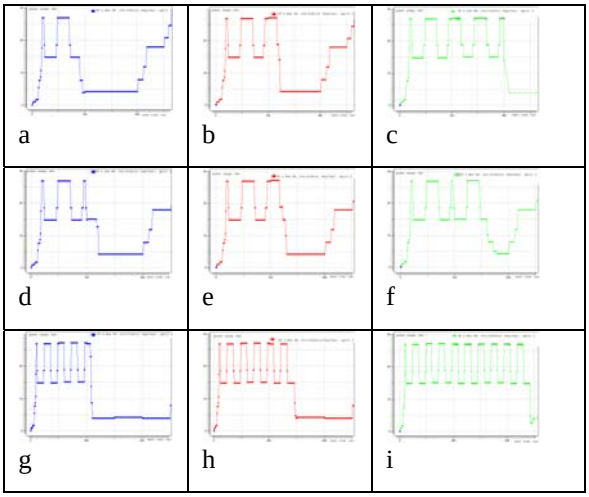
In the next section, we will compare the performance results of implemented DF a New PH algorithm with different parameters. To see which parameters provide the best performance, we will show only the first activity, *Executing the algorithm*, and the *sum* of power usage of three devices (the sum of CPU, Memory, and Radio). And also we will determine the life time of the sensor nodes and the aggregator node.

4.1 Different DF a New PH parameters in the Sensor Node Simulation.

We implement three different parameters of DF a New PH with twenty one alternatives. The first is nine alternatives of *Different Key Sizes and Fixed Message Sizes*. The second is *Different Message Sizes and Fixed Key Sizes*, also with nine alternatives. The third is *Different Message Split and Fixed Message Size*, with three alternatives.

4.1.1 Different Key Sizes, Fixed Message Sizes, Message Split 2

In this implementation, we are changing the key size parameter while keeping the message size fixed.



a. 128/128bits (msg size/key size), b. 128/256bits, c.128/512bits,
d. 256/256bits, e. 256/512bits, f. 256/1024bits,
g. 512/512bits, h. 512/1024bits, i. 512/2048bits.

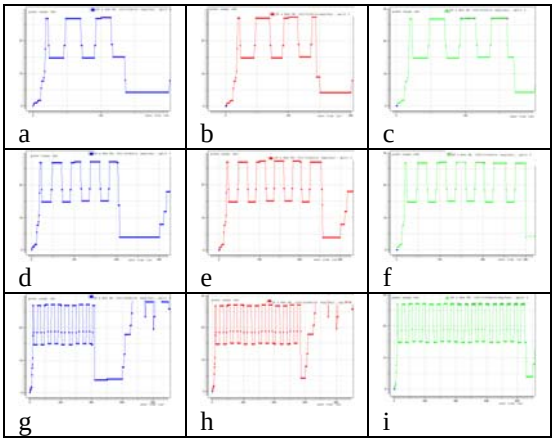
Figure 4: Performance results of DF a New PH with different key sizes

In terms of total power usage, the lowest is 128/128 (msg/key - figure 4a) and the highest is 502/2048

(msg/key – figure 4i). Above results confirm that the longer the size of key and message, the longer the sensor node to complete its cycle of operation.

4.1.2 Different Message Sizes, Fixed Key Sizes, Message Split 2

In this implementation, we are changing the message size parameter while keeping the key size fixed.



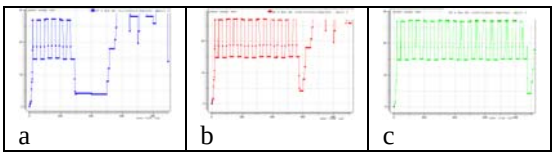
a. 100/512bits (msg size/key size), b. 250/512bits, c. 500/512bits, d. 250/1024bits, e. 500/1024bits, f. 1000/1024bits, g. 500/2048bits, h. 1000/2048bits, i. 2000/2048bits.

Figure 5: Performance results of DF a New PH with different message sizes

In terms of total power usage, the lowest is 512/100 (key/msg - figure 5a) and the highest is 2048/2000 (key/msg – figure 5i). Results confirm that the longer the size of key and message, the longer the sensor node to complete its cycle of operation.

4.1.3 Different Message Split (Split 3), Fixed Message Size, Different Key Sizes

In this implementation, we are changing the message split parameter to 3 splits while keeping the message size fixed.



a. 512/512bits (msg size/key size), b. 512/1024bits, c. 512/2048bits

Figure 6: Performance results of DF a New PH with message split 3

When we compare the same message/key size between different splits (Figures 4g-4i and Figures 6a-6c), by increasing the message split to 3, the sensor node requires more resources (longer CPU active, more Memory's READs and WRITEs) to accommodate bigger data processing.

Across the board, the best performance is on the fixed message size 128/128bits (msg/key – figure 4a) and the worst performance is tied between the fixed key size 2000/2048bits (msg/key – figure 5i) and the message split 3 on 512/2048bits (msg/key – figure 6c).

The higher the power consumed the shorter the sensor node's lifetime is. For illustration, on the sampling time interval of once per 1 minute (the sensor is active once per 1 minute and SLEEP until the next cycle), the sensor node of fixed message size 128/128bits (msg/key – Figure 4a) can survive for 14.39 days. While the fixed key size 2000/2048 (msg/key – Figure 5i) can only survive for 4.72 days. And on the message split 3, 512/2048 (msg/key – Figure 6c), the node survives for 4.75 days.

If we increase the sampling time interval, we get a substantial improvement of node lifetime. For example, if we increase sampling time interval from once/1minute to once /4minute, sensor node of 128/128bits (Figure 4a) lasts 230.23 days, while 2000/2048bits (Figure 5i) survive for 75.53days. And on the message split 3, 512/2048 (msg/key – Figure 6c), the node survives for 75.93 days.

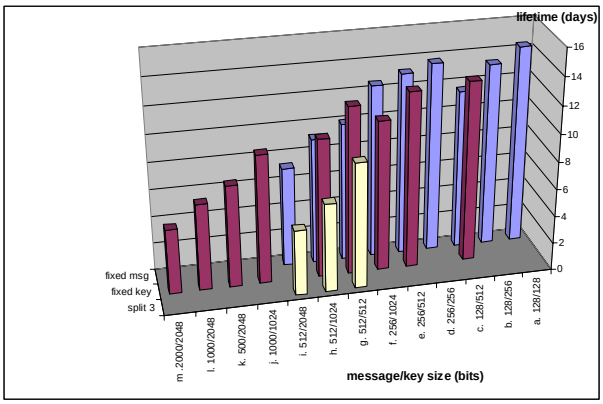


Figure 7: Node lifetime (in days) of DF a New PH implementation

Figure 7 shows the node's lifetimes of DF a New PH implementation, the back row (blue boxes) is on different key sizes and fixed message size, the middle row (purple boxes) is on different message sizes and fixed key size, and the front row (yellow boxes) is on message split 3.

When we increase the key size to four times of the message size (from 512/512 to 512/2048bits, split 2 Figure 7g to 7i), the sensor node lifetime decreases, approximately 27.9%.

This decrease is much worse on the message split 3. For example, with the same parameters, the node

lifetime decreases by 48%. As a result, we expect substantially shorter lifetimes of sensor nodes for message split more than 3. The bigger the message splits, the more resources that the sensor node uses to accommodate bigger data processing.

Since the Aggregator node only computes (addition) on the encrypted values and no decryption or encryption process is required, in this simulation, we also found that the Aggregator node performance is not related to encryption algorithm performance on the sensor nodes. On the other hand, its performance is determined by the number of sensor nodes which reports back to it. For instance, on cluster of 5 sensor nodes, the aggregator node can survive for 9.61 days, as opposed to a 20 sensor nodes cluster survives for only 3.42 days.

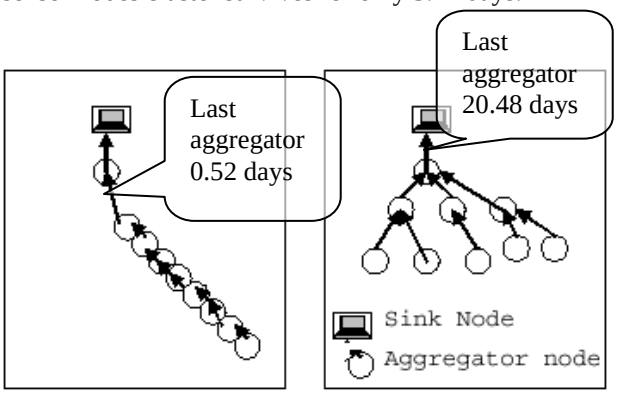


Figure 8: The last aggregator node lifetime (in days) of DF a New PH implementation

Figure 8 shows the large scale network level simulation model which consists of 1000 clusters (total: 11000 sensor nodes). In this level, sampling time interval is once/4minute and all sensor nodes perform DF a New PH with split 2, 512/512bits (msg/key) parameters. Sensing area is 500 x 500 square meters. The network configuration is based on Djikstra's Shortest Open Path Algorithm[27]. We collected information about the life time of the last aggregator node which is an aggregator node directly connected to the sink node based on two network configurations. In the first configuration where all clusters formed a single line – Figure 8, left, and in the second configuration where all clusters formed a balanced tree – Figure 8, right. The last aggregator node that directly connected to the Sink node in first configuration can survive for 0.52 days and 20.48 days in second configuration. This result shows that network configuration rather than the security algorithm that is used for end-to-end encryption affects the life time of the last aggregator node.

5 Conclusion

In this paper, we have shown that the DF a New PH is feasible to be implemented on MICA2™ sensor nodes. We conclude that the shorter the key and the message sizes will yield the better sensor node performance, while the longer the key and message sizes with the bigger message split will yield poor performance and shorter sensor node lifetimes.

While more message splits offer stronger security level, it has the consequence of performance penalty therefore the message split should be kept at minimum. To balance the acceptable level of service and level of security, other parameters should be selected based on desired performance and the lifetime of sensor nodes.

There are other potential HES such as DF Allowing Field Operations, DF Additive & Multiplicative and Mixed Multiplicative (MMH) available [28] [29] [30]. These algorithms could also be feasible to be implemented on WSN. In the future, we are planning to implement these algorithms to find best possible HES for providing WSN security.

7 References

[1] Y. Xiao, "Security in Sensor Networks," *Auerbach Publications*, pp. 275-290, 2007.

[2] D. Integrity, P. Sakarindr, and N. Ansari, "Security Services IN Group Communications OVER Wireless Infrastructure, Mobile Ad Hoc, AND Wireless Sensor Networks," *IEEE Wireless Communications*, pp. 9, 2007.

[3] B. Sun, L. Osborne, Y. Xiao, and S. Guizani, "Intrusion detection techniques in mobile ad hoc and wireless sensor networks," *Wireless Communications, IEEE [see also IEEE Personal Communications]*, vol. 14, pp. 56-63, 2007.

[4] J. C. Lee, V. C. M. Leung, K. H. Wong, J. Cao, and H. C. B. Chan, "Key management issues in wireless sensor networks: current proposals and future developments," *Wireless Communications, IEEE [see also IEEE Personal Communications]*, vol. 14, pp. 76-84, 2007.

[5] K. Jones, A. Wadaa, S. Olariu, L. Wilson, and M. Eltoweissy, "Towards a new paradigm for securing wireless sensor networks," *Proceedings of the 2003 workshop on New security paradigms*, pp. 115-121, 2003.

[6] J. Girao, D. Westhoff, M. Schneider, N. E. C. E. Ltd, and G. Heidelberg, "CDA: concealed data aggregation for reverse multicast traffic in wireless sensor networks," *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, vol. 5, 2005.

[7] M. Acharya, J. Girao, and D. Westhoff, "Secure Comparison of Encrypted Data in Wireless Sensor Networks," *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, 2005. WIOPT 2005. Third International Symposium on*, pp. 47-53, 2005.

[8] D. Westhoff, J. Girao, and M. Acharya, "Concealed Data Aggregation for Reverse Multicast Traffic in Sensor Networks: Encryption, Key Distribution and Routing Adaptation," *IEEE Transactions on Mobile Computing, October*, 2006.

[9] D. Westhoff, J. Girao, and A. Sarma, "Security Solutions for Wireless Sensor Networks," *Nec Technical Journal*, vol. 1, 2006.

[10] E. Vaidehi, "*Computing Aggregation Function Minimum/Maximum using Homomorphic Encryption Schemes in Wireless Sensor Networks (WSNs)*," 2007.

[11] T. Sander and C. F. Tschudin, "Protecting Mobile Agents Against Malicious Hosts," *Mobile Agents and Security*, vol. 60, 1998.

[12] A. Varga, "OMNeT++ User Manual," *Department of Telecommunications, Technical University of Budapest*, 1997.

[13] B. Stroustrup, "The C++ Programming Languge," 1997.

[14] I. C. Technology, "MICA2: Wireless Measurement System," *Mica2 Datasheet. Available in: http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet. pdf.*

[15] M. Yokoo and K. Suzuki, "Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auctions," *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 112-119, 2002.

[16] Domingo-Ferrer, "A new privacy homomorphism and applications," *Information Processing Letters*, vol. 60, pp. 277-282, 1996.

[17] W. Stallings, "Cryptography and Network Security: Principles and Practice," 2006.

[18] C. Ding, D. Pei, and A. Salomaa, "Chinese remainder theorem," 1996.

[19] C. Mallanda, A. Suri, V. Kunchakarra, S. S. Iyengar, R. Kannan, and A. Durresi, "Simulating Wireless Sensor Networks with OMNeT++," *Dept. of Computer Science, Louisiana State Univ. Retrieved*, vol. 9, pp. 2005, 2005.

[20] K. Pawlikowski, H. D. J. Jeong, and J. S. R. Lee, "On credibility of simulation studies of telecommunication networks," *Communications Magazine, IEEE*, vol. 40, pp. 132-139, 2002.

[21] S. Roundy, D. Steingart, L. Frechette, P. Wright, and J. Rabaey, "Power Sources for Wireless Sensor Networks," *Energy (J/cm 3)*, vol. 3780, pp. 1200, 2004.

[22] V. Shnayder, M. Hempstead, B. rong Chen, and M. Welsh, "PowerTOSSIM: Efficient power simulation for TinyOS applications," *Proc. SenSys*, 2004.

[23] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 188-200, 2004.

[24] C. Negus, "Linux Bible: Boot Up to Fedora, KNOPPIX, Debian, SUSE, Ubuntu, and 7 Other Distributions," 2006.

[25] G. Bianchi, L. Fratta, and M. Oliveri, "Performance Evaluation and Enhancement of the CSMA/CA MAC Protocol for 802.11 Wireless LANs," *Proc. PIMRC*, pp. 392-396, 1996.

[26] M. S. Gast, "802.11 Wireless Networks: The Definitive Guide," 2002.

[27] B. Liu, S. H. Choo, S. L. Lok, S. M. Leong, S. C. Lee, F. P. Poon, and H. H. Tan, "Finding the shortest route using cases, knowledge, and Djikstra'salgorithm," *Expert, IEEE [see also IEEE Intelligent Systems and Their Applications]*, vol. 9, pp. 7-11, 1994.

[28] J. Domingo-Ferrer and J. Herrera-Joancomarti, "A privacy homomorphism allowing field operations on encrypted data," *I Jornades de Matematica Discreta i Algorismica, Universitat Politecnica de Catalunya*, 1998.

[29] J. Domingo-Ferrer, "A Provably Secure Additive and Multiplicative Privacy Homomorphism," *Information Security Conference*, pp. 471–483, 2002.

[30] H. Lee, J. Alves-Foss, and S. Harrison, "The use of encrypted functions for mobile agent security," *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, pp. 297-306, 2004.