Secure and Efficient Sharing Aggregation Scheme for Data Protection in WSNs

Feriel Bouakkaz, Mawloud Omar, Ahcène Bounceur, Abdelkamel Tari

Secure and Efficient Sharing Aggregation Scheme for Data Protection in WSNs

Feriel Bouakkaz[1], Mawloud Omar[1], Ahcène Bounceur[1,2] and Abdelkamel Tari[1]

[1]Laboratoire d'Informatique Médicale, Faculté des Sciences Exactes
Université de Bejaia, 06000 Bejaia, Algérie.
[2]Lab-STICC - UMR CNRS 6285
Université de Bretagne Occidentale, 29238 Brest, France.

Email: bouakkazferiel@yahoo.fr

***Abstract*—Wireless sensor networks (WSNs) are omnipresent in a multitude of applications. One of the important common requirements of these applications is the data security. Indeed, the exchanged data in WSNs are often considered as a preferred target, which can be a subject of several threats, such as eavesdropping, replay, falsification, alteration, etc. Another important common requirement of WSNs applications is data aggregation. Indeed, the limitations of such networks in terms of energy, bandwidth and storage accentuate the need of data aggregation. In this paper, we address these two issues. We propose a new efficient approach for data integrity and credibility protection for WSNs, while ensuring the data aggregation. We consider a cluster-based network architecture, where sensor nodes are equally distributed in clusters. Each sensor node is in charge to deliver one bit of the sensed data and at the same time observe the remaining parts through a parity control based encryption approach. In this manner, the sensed data could be effectively and securely controlled with a low overhead compared to the classical aggregation approaches, where all the nodes transmit individually the sensed data. To validate the proposed protocol we have simulated it using the simulator CupCarbon and in order to evaluate its efficiency in terms of energy, we have developed a prototype with the TelosB platform, where the obtained results show that our method is less energy consuming.[1]**

I. Introduction

The sensors have much seduced by their many advantages. They very quickly invade the field of medicine, industry, environment, military and many other fields. These equipments are of small size, low price, have the ability to capture different information (thermal, optical, vibration, etc.) and can communicate wirelessly. To benefit from these advantages possessed by the sensors, they are deployed in large number in areas of interest forming a WSN to perform a common task [2][5]. However, WSNs suffer from energy problem, because it is difficult to change the batteries of sensors after their deployment. In other hand, WSNs suffer from vulnerability problem, due to the nature of wireless communication. These constraints highlight the difficulty of developing the necessary protocols for the management and protection of these networks, such as routing and security protocols. As the other types of network architectures, security is one of the most challenging requirements in WSNs [11] and includes mainly data confidentiality, data integrity, authentication, availability and data freshness. In the literature, the majority of works uses the symmetric cryptography. The latter takes into account the constraints of this network type, nevertheless the key distribution stays a challenge. Asymmetric cryptography is more secure than symmetric one, but is more expensive in term of resources.

There are numerous works using the threshold cryptography: a number of nodes cooperate in order to create the secret, and an attacker has to successfully compromise these nodes to alter the data exchanged. Their results are very promising. Shen et al. [9] have proposed a protocole based on μTESLA [13] and threshold cryptography in order to provide a broadcast authentication for multi base stations sensor networks. The proposed protocol divides the authentication key into key shares, and distributes them to each station. Sensor nodes reconstruct keys by using key shares broadcasted by Base station and authenticate the broadcast messages. Li et al. [7] have proposed a secure monitoring scheme using identity-based threshold signcryption scheme. The proposed protocol provides a confidential and authenticated transmission channel for monitoring messages sent by the sensor nodes to the base station. Sliti et al. [8] have proposed a technique called: "k-security in heterogeneous WSNs" based on elliptic curve cryptography and threshold signature. A minimal number of k sensor nodes is required to individually sign a message in order to generate a global valid signature. It requires that an alert message related to a hostile presence within the monitored region should be issued by k distinct sensor nodes in order to be considered as valid. Singh et al. [3] have proposed a hierarchical group key management using threshold cryptography for WSNs. The technique considers hierarchical sensor network, where sensing nodes are coordinated by forwarding nodes. The latter nodes are connected to the base station and are

[1]This work is part of the research project PERSEPTEUR supported by the French Agence Nationale de la Recherche ANR.

responsible for key computation and distribution. A forwarding node computes the group key using a threshold secret sharing scheme. The acquired group key is divided into multiple shares and shared among the member nodes. Koschuch et al. [6] have proposed an adaptation of the multiparty multiplication protocol for cluster-based WSNs. The authors have focused on the number of exchanged messages and the generated overhead on each sensor node. The secret is signed by the sensor nodes by using RSA, and it is sent to the cluster head in order to calculate the complete valid signature. Chaudhary et al. [1] have proposed a group authentication scheme for WSNs using threshold cryptography. It is used to authenticate multiple devices at once. The group manager and the members sharing a pair of RSA keys used to exchange a message. This message allows devices to get partially decrypted message (PDM). All of these PDMs are sent to the groupe manager in order to recover the complete message and to authenticate the group.

In this paper, we propose a new protocol that aims to ensure two important functions in the operation of WSNs: the protection of data integrity and the aggregation. This protocol is based on threshold cryptography, which consists of the collaboration of sensor nodes in order to deliver collectively the sensed data. Each sensor node proceeds to the aggregation by sending to the cluster head a part of the sensed value and control bits. The latter bits are used by a system of parity on different bit combinations of the message. A cluster head, upon receiving all the parts, recovers the complete value, verifies its integrity by using the control bits and corrects the message in case of attacks. To validate the proposed method we have simulated it using the simulator CupCarbon [14] and in order to evaluate its efficiency in terms of energy, we have developed a prototype with the TelosB platform.

The remainder of this paper is organized as follows. In Section II, we give the detailed description of our protocol, as well as we discuss the resistance of our protocol against attacks. In Section III, we present the simulation results of our protocol using the simulator CupCarbon and its implementation on real sensors. These results have allowed us to validate our protocol and study its effectiveness in terms of energy consumption. Finally, in Section IV, we conclude the paper.

II. The proposed protocol

In this section, we present the considered network model, the operations of the proposed protocol and the security analysis.

A. Network model and assumptions

We consider a cluster-based WSN composed of a set of sensor nodes deployed in a zone of interest. We assume that sensor nodes are static, their wireless mediums are error free, and have all the same hardware characteristics. The network is divided into several clusters with the same node number, and where each cluster is supervised by a cluster head. We assume that the sensor nodes belong to the same cluster are geographically close so that the sensed data will be the same for each sensor node of the same cluster. Several WSN applications operate under this property. For example, in wireless body area networks, the sensor nodes scratched on a same patient's body observe approximately the same glucose state. The sensor nodes deployed in a volcanic area observe approximately the same temperature if the nodes are geographically close. In military applications, an adversary is detected simultaneously by sensor nodes supervising the same area. We note n the number of the sensor nodes that belong to each cluster. In order to provide the cluster size balance by adjusting n, we can use the potential-based clustering protocol, proposed in [10]. This protocol uses *Hello* packets to estimate the potential "number of neighbors" of each node. Each node maintains the potentials of its neighbors and determines the budget amount assigned to each one of them. A neighbor with higher potential receives more tokens than a neighbor with a lower potential. The same distribution process is applied by sensor nodes in the subtrees until the budget is exhausted or no further growth is possible. Let consider the binary format of the sensed environment parameter $V = (v_1, v_2, \cdots, v_n)$, where n is equal to the number of the bits of V (i.e., $n = |V|$ bits) and $v_i \in \{0, 1\}$. In Table I, we summarize the used notations in this paper, where $i = 1, ..., n$.

TABLE I. NOTATIONS

Notation	Description
n	The cluster size ($n = \|v\|$)
S_i	The i^{th} sensor node
V	The sensed data $v = (v_1, v_2, \cdots, v_n)$
V'	The reconstructed sensed data
m_i	The message sent by S_i
m_i'	The message received by the cluster head
r	The number of control bits
C	The control bit (parity bit) matrix
G	The generator matrix

B. The protocol algorithm

Before the network deployment, each sensor node is pre-configured with a set of symmetric keys shared with all the other sensor nodes of the network. These keys are used to secure the communication between sensors, cluster heads and the base station. After deployment, excepting the cluster heads, each sensor node discovers its neighbors, identifies the required symmetric keys to be used. However, cluster heads keep in addition, symmetric keys of the sensor nodes that belong to their cluster. The main purpose using these symmetric keys is to secure multi-hop communication when transmitting sensitive information as parts of the sensed value and the exchanged data between the cluster heads and the base station. Other operations to do before the deployment are the computation of the control bit number r and the generation of the $r \times n$ binary matrix G given by Equation (1), and where the rows represent a succession of 2^i of 1 and 0, where $i = 1, ..., n$ is the column number in G.

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & . \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & . \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & . \\ . & . & . & . & . & . & . & . & . \end{bmatrix} \quad (1)$$

The parameter r depends on the size of the sensed data and is calculated by solving the following equation:

$$n = 2^r - y \quad (2)$$

where y is a positive variable that must be minimized.

Once an event is detected, each sensor node S_i extract from the sensed data V its i^{th} bit, where i corresponds to the identifier of S_i which is related to its order in the cluster. Then, it calculates the control bits $c_1, c_2, ..., c_r$, where c_i represents the even parity of the vector $b_i = (g_i \& v)$, g_i is the i^{th} row of G and & represents the logical AND operator. Figure 1 illustrates how to calculate c_j for $n = 5$ and $r = 3$.

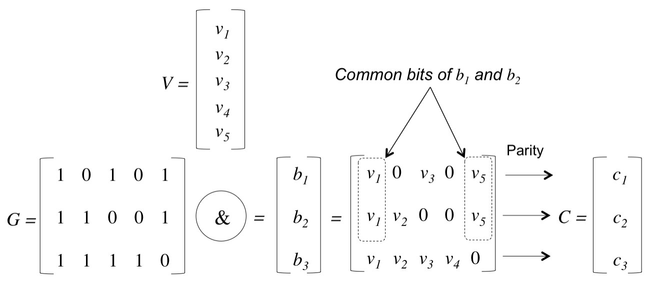


Fig. 1. Computation of the control bits c_j for $n = 5$.

A message m_i sent by the sensor node S_i to the cluster head is composed of v_i, c_i^1, ..., c_i^r. That is to say, $m_i = [v_i|c_i^1|c_i^2|...|c_i^r]$. The character $'|'$ represents the conactenation, v_i is the i^{th} bit of V sent by the sensor node S_i and c_i^j is the j^{th} control bit sent by the sensor node S_i. The control bits are used to verify the integrity of the message. Note that the message m_i is sent once encrypted. The reconstruction of the message by the cluster head starts by the decryption of all the messages m_i' received from its neighbors, where $m_i' = [v_i'|c_i'^1|c_i'^2|...|c_i'^r]$. The next steps are; concatenates v_i', uses the control bits to verify the integrity of the message and transmits it securely to the base station. Algorithm 1 and Algorithm 2 shows the pseudo code of the proposed protocol to implement on each sensor node and each cluster head.

Algorithm 1 The pseudo code of a cluster head.

```
Input: $n$: The number of the sensors of the cluster
 1: Calculate the value of $r$
 2: Generate the matrix $G$
 3: repeat
 4:     Read/Decrypte the $n$ received messages: $m_i' = [v_i'|c_i'^1|c_i'^2|...|c_i'^r]$, $i = 1, ..., n$
 5:     Reconstruct the sensed value $V' = [v_1'|v_2'|...|v_n']$
 6:     Calculate the vector $C = (c_1, c_2, ..., c_r)$
 7:     Compare each received $C_i' = (c_i'^1, c_i'^2, ..., c_i'^r)$ to $C$
 8:     if $(\forall k / C \neq C_k')$ then
 9:         Determine the malicious node and correct $V'$
10:     end if
11:     Send $V'$ to the base station
12: until false
```

Algorithm 2 The pseudo code of a sensor node.

```
Input: $n$: The number of the sensors of the cluster
Input: $i$: The identifier of the current sensor node
 1: Calculate the value of $r$
 2: Generate the matrix $G$
 3: repeat
 4:     Wait for a message from the cluster head
 5:     Read the sensor value $V$ (a binary format)
 6:     Extract the $i^{th}$ bit of $V$
 7:     Calculate the vector $C = (c_1, c_2, ..., c_r)$
 8:     Construct the message $m_i = [v_i|c_i^1|c_i^2|...|c_i^r]$
 9:     Encrypt $m_i$
10:     Send $m_i$ to the cluster head
11: until false
```

C. Behavior against attacks

If an attacker, internal or external, tries to alter the exchanged data, it will attempt to modify the information while ensuring that it will not be detected. In front of this type of attack, the proposed protocol can detect this modification, and can also identify the malicious node, with the possibility to correct the message. In our protocol, each node sends a bit of the message, which limits the power of each node to alter the entire data. Hence, any malicious node can only modify one bit of the message. This modification will be immediately detected by the cluster head through the control bits. These bits represent the bit parity of the received message V'. If any malicious node change some of the bits of the sent message V then this parity will not be longer verified. Therefore, the attack will be detected. The attack detection phase will be followed by the identification of the malicious nodes phase. This will be done by identifying the common bit of the incorrect control bits. We will come back again to the situation illustrated by Figure 1. In this case, if the parity is not verified by the control bits c_1' and c_2' we start first by identifying the common bits (v_1' and v_5' in our example). Then, from these common bits, we will determine those that are not verified by c_3' (v_5' in our example). This allows as to determine the malicious nodes, where the i^{th} node is declared as malicious if the corresponding i^{th} bit is not verified by c_3' (i.e., the fifth node is malicious in our example). By analyzing the behavior of the proposed protocol against active attacks, we conclude that if any attacker try to modify any bit of the message V without being detected, it must modify this bit in at least $(n/2) + 1$ nodes.

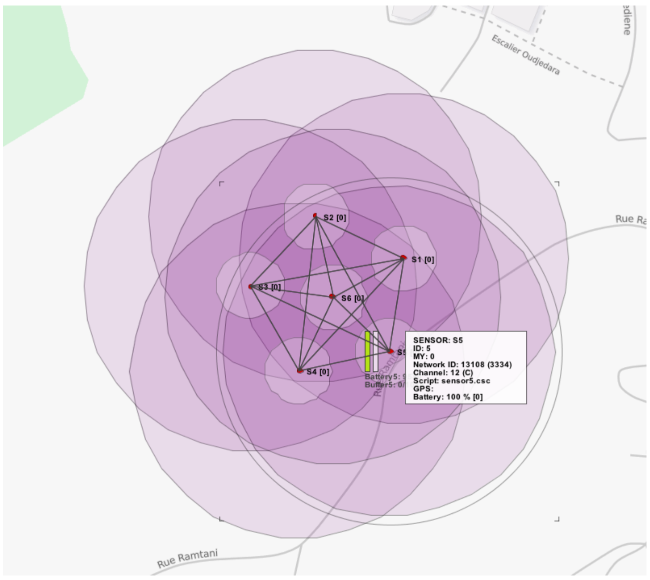
III. Simulation and Implementation

To validate the proposed protocol, first, we have simulates it using the CupCarbon platform [14][15] and then we have implemented it using real sensor nodes (TelosB).

A. *Simulation results*

The simulations are done using the CupCarbon simulator, which is a Smart City and Internet of Things Wireless Sensor Network (SCI-WSN) simulator. Its objective is to design, visualize, debug and validate distributed algorithms for monitoring, environmental data collection, etc. and to create environmental scenarios such as fires, gas, mobiles. Networks can be designed and prototyped by an ergonomic and easy to use interface using the OpenStreetMap (OSM) framework to deploy sensors directly on the map. It includes a script called SenScript which allows to program and to configure each sensor node individually.

Figure 2 shows a network with 6 nodes (one cluster head S_6 and five sensor nodes S_1, S_2, S_3, S_4 and S_5) designed using the simulator CupCarbon.



SENSOR: S5
ID: 5
MY: 0
Network ID: 13108 (3334)
Channel: 12 (C)
Script: sensor5.csc
GPS:
Battery: 100 % [0]

Fig. 2. Network designed using the CupCarbon simulator.

We have validated that the proposed protocol can detect malicious nodes and correct the switched bit of the received data. In the simulations we assumed that the energy consumption related to the the computing operations is zero. To estimate the energy consumption, we use the energy model of the TelosB. It is estimated to $59.2\mu J$ to transmit one bit and to $28.6\mu J$ when one bit is received [16]. We have used the Super Alkaline AALR6 battery which is a portable energy source with a capacity of 9580 Joules.

Figure 3 shows the state of the battery of each sensor node in the case where the cluster head asks four times for the sensed value each 1 seconds. We have considered a data with 5 bits. Note that the sensor node S_6 is the cluster head.

Figure 4 shows the energy consumption (in Joules) of the sensor node S_1 in function of the time (in seconds). If we look the zoomed part of the peak, we can see two peaks that are generated by the received message from the cluster head that

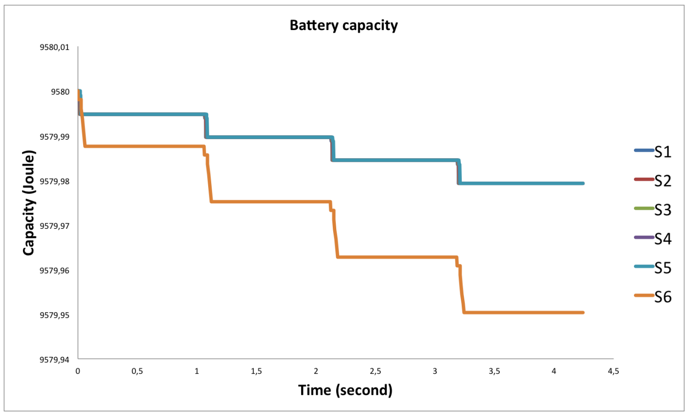


Fig. 3. The state of the battery vs. the simulation time.

asks for the sensed data (the small peak) and by the message to send the sensed data to the cluster head. We conclude that, the consumption of a sensor is mainly due to the sending of the sensed data.

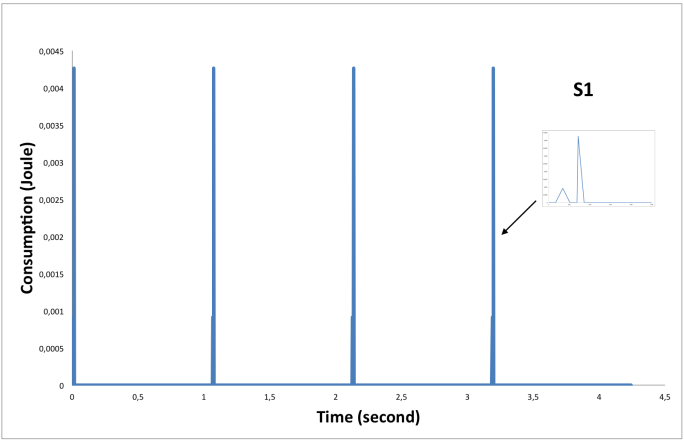


Fig. 4. The energy consumption vs. the simulation time.

B. *Real implementation*

Figure 5 shows a MTM-CM5000-MSP which is the sensor nodes used in our experiment.

It is IEEE 802.15.4 compliant wireless sensor node, based on the original open-source ”TelosB/Tmote Sky platform design” developed and published by the University of California, Berkeley. The included sensors are: temperature, relative humidity and light. To approach has been implemented using the Contiki OS [12]. It is an open source and portable operating system designed specifically for resource limited devices such as sensor nodes. It brings the benefits of both events and thread execution models. It also supports a full TCP/IP stack via uIP and the programming abstraction ”Protothreads”. The Contiki-OS version which we have used is InstantContiki2.7-VMware Player.

We have developed two prototypes: one in the cluster head side and the other in the sensor side according to the

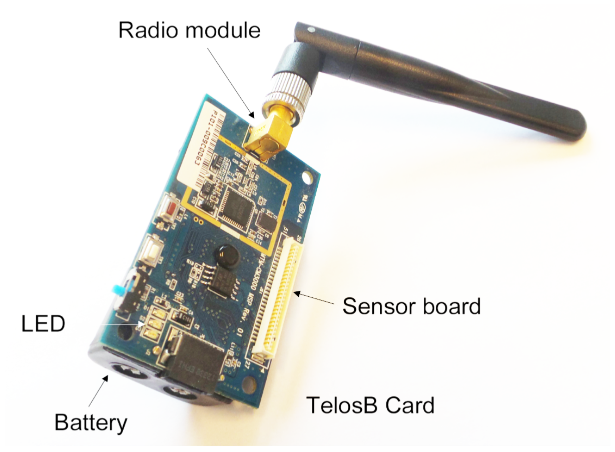


Fig. 5. A TelosB sensor node.

steps presented previously. These programs are developed in language C, injected into the sensor nodes by using the virtual machine of Contiki-OS. To compare the simulation results with the real ones, we have designed the same network of Figure 2 as illustrated by Figure 6. The cluster head is connected to a laptop to monitor in real time the execution of the protocol and collects data via the USB port by using the virtual machine of Contiki-OS.

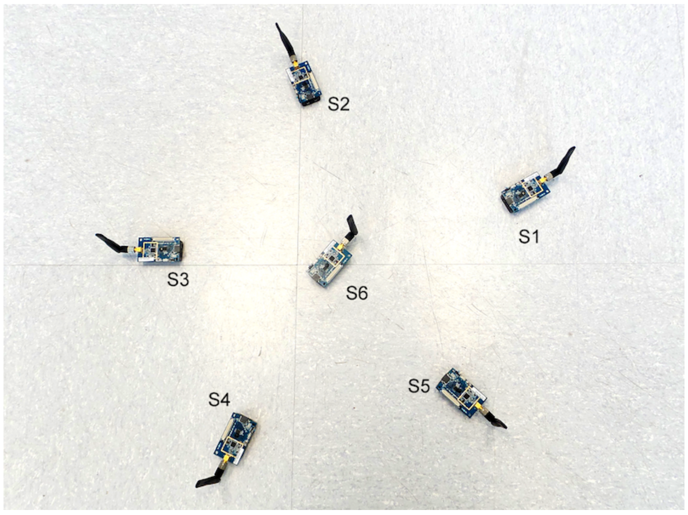


Fig. 6. Network deployment.

To profile the power consumed by the sensors, we have used the Powertrace system [4] which gives values that are very close to the real ones. Powertrace calculates the power consumption of the local node based on the monitoring of the power state. This value is then encapsulated according to the corresponding activities (reception or transmission of packets, computation, etc.). The energy consumption is computed in Joules as:

$$E = \frac{\mathrm{EnergestValue} \times \mathrm{Current} \times \mathrm{Voltage}}{\mathrm{RtimerSecond} \times 1000} \tag{3}$$

where, $\mathrm{Current} = 20\mathrm{mA}$ for listening, $17.7\mathrm{mA}$ for transmitting and $1.8\mathrm{mA}$ for computing. $\mathrm{Voltage} = 3\mathrm{V}$ and $\mathrm{RtimerSecond} = 32768$. Energy consumption profiling of our protocol is illustrated by Figure 7.

The blue curve, in the upper part, represents the energy consumption generated by sending four times the value of the temperature sensor. This means that our protocol will be executed four times by each sensor. While the lower part shows the corresponding activities for each consumption. These two parts have been obtained using the Powertrace system. This graph shows that the power consumption is close to 0.001 Joules. This consumption is generated by passive monitoring operations, the execution of Powertrace and other network management operations running continuously. However, the four peaks are due to energy consumed by three activities: the reception activity (in green color), the transmission of data to the cluster head (in blue color) and the computation activity (in red color). If we compare this figure with Figure 4 we can see that the results obtained by simulation and using real sensor nodes are close. Then, we can conclude that the proposed protocol is less energy consuming and it is estimated at 0.0048 Joules to execute one time the proposed protocol. Using the Super Alkaline AALR6 battery with a capacity of 9580 Joules, our protocol can be executed $\sim 2 \cdot 10^6$ times.

IV. Conclusion

In this paper, we have proposed an efficient shared protocol that ensures the data integrity. In this protocol, each sensor node transmits a part of the sensed data and in the same time observes the remaining parts through a parity control-based encryption approach. The cluster head reconstitutes the data by concatenating the received parts, uses the control bits to detect and correct the message in case of attack, and finally forwards it securely to the base station. We proved the robustness of our protocol by studying its behavior against attacks. Subsequently, we validated it by simulation using the CupCarbon simulator and then implemented it in a real environment based on the TelosB sensor nodes. The obtained results in terms of energy consumption show that the proposed protocol is less energy consuming and can be used to send about $2 \cdot 10^6$ of 5 bit coded sensed value.

References

[1] K. Chaudhary and G. Shinde. *Group Authentication in Wireless Sensor Networks*. International Journal of Scientific Engineering and Research, 2015.

[2] H.M. Ammari. *The Art of Wireless Sensor Networks: Volume 1: Fundamentals*. Springer Science and Business Media, 2013.

[3] K. Singh and L. Sharma. *Hierarchical Group Key Management using Threshold Cryptography in Wireless Sensor Networks*. International Journal of Computer Applications, 2013.

[4] A. Dunkels, J. Eriksson, N. Finne, and N. Tsiftes. *Powertrace: Network-level Power Profiling for Low-power Wireless Networks*. SICS Technical Report T2011:05 ISSN 1100-3154, 2011.

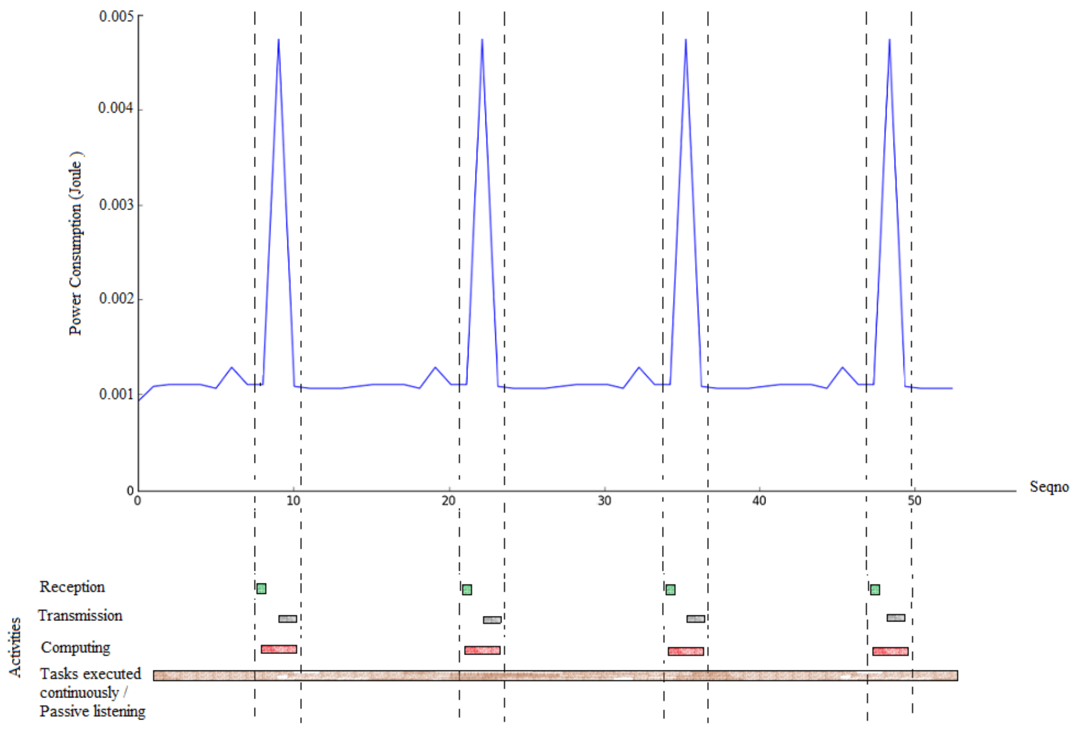


Fig. 7. Energy consumption profiling of our protocol.

[5] I.F. Akyildiz and M.C. Vuran. *Wireless Sensor Networks*. Advanced Texts in Communications and Networking, Wiley, 2010.

[6] M. Koschuch, M. Hudler, M. Kruger, P. Lory, and J. Wenzel. *Applicability of multiparty computation schemes for wireless sensor networks*. Proceedings of the Data Communication Networking, Athens, 2010.

[7] J.F. Li, D.W. Wei, and H.Z. Kou. *Secure Monitoring Scheme Based on Identity-Based Threshold Signcryption for Wireless Sensor Networks*. Proceedings of the 4th Wireless Communications, Networking and Mobile Computing, 2008.

[8] M. Sliti, M. Hamdi, and N. Boudriga. *An Elliptic Threshold Signature Framework for k-Security in Wireless Sensor Networks*. Proceedings of 15th Electronics, Circuits and Systems, St. Julien's, 2008.

[9] Y.L. Shen, Q.Q. Pei, and J.F. Ma. *MMμTESLA: broadcast authentication protocol for multiple-base-station sensor networks*. Chinese Journal of Computers, 2007.

[10] F.B. Abdesslem, A. Ziviani, M. Dias de Amorim, and P. Todorova. *Looking Around First: Localized Potential-Based Clustering in Spontaneous Networks*. Communications Letters, 2007.

[11] N.R. Prasad and M. Alam. *Security Framework for Wireless Sensor Networks*. Journal of Wireless Personal Communications, 2006.

[12] A. Dunkels, B. Gronvall, and T. Voigt. *Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors*. In first Workshop on Embedded Networked Sensors, 2004.

[13] A. Perrig, R. Szewczyk, J.D. Tygar, V. Wen, and D.E. Culler. *SPINS: security protocols for sensor networks*, Wireless Networks Journal, 2002.

[14] K. Mehdi, M. Lounis, A. Bounceur, and T. Kechadi, Cupcarbon: A multi-agent and discrete event wireless sensor network design and simulation tool, *In IEEE 7th International Conference on Simulation Tools and Techniques (SIMUTools'14)*, Lisbon, Portugal, March 17-19, 2014.

[15] CupCarbon. (2015) ANR Project PERSEPTEUR, CupCarbon simulator. [Online]. Available:http://www.cupcarbon.com

[16] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, Energy analysis of public-key cryptography for wireless sensor networks, In the Third IEEE International Conference on Pervasive Computing and Communications, PerCom 2005, pp. 324328.