Energy-Driven Methodology for Node Self-Destruction in Wireless Sensor Networks

Madalin Plastoi, Daniel-Ioan Curiac
"Politehnica" University Timisoara, Romania
madalin.plastoi@aut.upt.ro, daniel.curiac@aut.upt.ro

Abstract— **Wireless sensor networks technology is a rapidly growing domain, getting more and more credit in the area of civilian and military applications. In the same time with technological advancement, new and dangerous information security threats have emerged. In this paper we considered that a node self-destruction procedure must be performed as a final stage in the sensor node lifecycle in order to assure the confidentiality regarding information like: network topology, type of measurement data gathered by sensors, encryption/authentication algorithms and key-exchange mechanisms, etc. that can be unveiled otherwise through reverse engineering methods. Our methodology relies on an efficient power monitoring scheme, based on combined in-network and predictive data, which discover the low battery nodes and initiate a self-destruction procedure for that nodes.**

I. INTRODUCTION

A wireless sensor network (WSN) is a collection of tiny devices (motes) having three main capabilities – sensing, processing and data routing. Each mote is equipped with: microcontroller for data processing, storage units, radio transceiver, power supplies and one or more dedicated sensors. Due to nodes physical dimensions and their wireless communication extension, wireless sensor networks could be used almost in any environment for a wide range of applications [1]. Beside WSN advantages, few important constraints still remain: limited energy and bandwidth, low computational capabilities and of course data security challenges.

In many cases, WSN deals with sensitive information that can represent an attractive target for potential attacks [2], [3]. Being deployed in harsh environments the lack of physical protection is a real issue. In these circumstances, network nodes might be captured by an attacker and used them, after reverse engineering techniques were applied, for malicious purposes – capturing measurement information, getting and reading network authentication keys and protocols, injecting bad information into the network, etc. Some of these threats can occur even if the WSN ended its operational phase of its lifecycle and the nodes are abandoned in the field. In this case, the attacker has access to a large quantity of nodes and through reverse engineering techniques he can obtain information regarding the reason of the WSN deployment in that area, the network topology, the type of measurement data gathered by sensors, the encryption/authentication algorithms and key-exchange mechanisms, etc. To avoid this type of attacks, a new strategy has to be developed. This paper solves this problem by destructing the low energy nodes using an energy-driven methodology.

Our strategy discovers the low battery nodes using an efficient energy power monitoring method. This method is developed for the base station level where enough computational power is available and will observe and predict network's energy evolution. If a low battery node is discovered the base station will take the decision to destruct it by activating a self-destruction code already placed in node's memory at a specified location.

II. NETWORK ASSUMPTIONS

Our methodology relies on a cluster-tree wireless sensor network topology [4]. Network has a high density of nodes (referred as motes) grouped in trees and forming a multi-hop communication environment. (Fig.1):

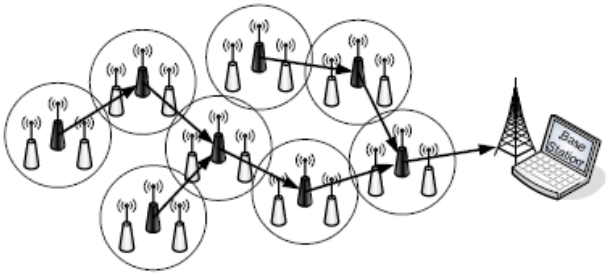

Figure 1. Clustered-based wireless sensor network

All network nodes belong to the MicaZ family of motes and implement the IEEE/ZigBee 802.15.4 standard [5], conducting to lower energy consumption for the entire network. Separately, the cluster head nodes have increased energy consumption, as they build the "backbone" of the network. The base station is considered to be a laptop class device.

III. PROPOSED METHODOLOGY

Our methodology is developed around the following scenario - we assume that in-network spread data represents a target for intrusion attacks. In order to countermeasure a hypothetic attack, we have to assure that every node with less energy than needed will be destroyed – we refer to an informational destruction and not necessarily to a physical one; otherwise, that node will be left outside of the operating network and could be easily captured and used for malicious purposes [6][7]. For preventing this event to happen, we suggest a strategy that comprises three main components, presented in Fig.2.

978-1-4244-4478-6/09/$25.00 ©2009 IEEE

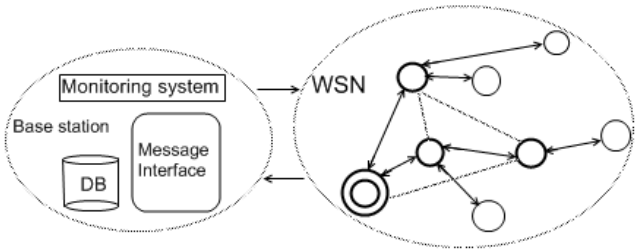

Figure 2. Components and architecture

These components are:

a) an energy monitoring system based on in-network information and prediction of the battery energy;
b) a decisional module that will identify the nodes with low power batteries;
c) a self-destructing distributed system that involves all wireless networks components (network coordinator, cluster-heads and ending notes).

Because our methodology is designed to be performed on the base station level, where enough computational power is available, we chose to make an implementation based on threads. Our methodology for a single network node is described by the following pseudocode (Fig. 3):

```
01.  START thread for node A
     {
02.    SET ε , τ , node_profile, destroy_indicator = 0;
03.    REPEAT at every τ seconds
       {
04.      READ  V_A(t) ;
           /*reads the sensor's battery voltage
05.      COMPUTE  V̂_A(t+1) ;
           /* computes  the predicted node  battery
           /* voltage using AR-prediction
06.      IF (( V_A(t) < ε ) or ( V̂_A(t+1) < ε )
         {
07.          SEND self-destructing activation message;
08.          destroy_indicator=1;
         }
       }
09.    UNTIL(destroy_indicator=1)
     }
```

Figure 3. Methodology pseudocode

The methodology starts with an initialization phase (line 02 from Fig. 3) that will set up some useful constant values: the *node_profile* which indicates if the node is a cluster-head node, an intermediary routing node, or a sensing node; a threshold ε representing the fraction of the highest battery voltage $V_{A,\max}$ below which the sensor node is considered to be useless and must be destroyed, a timestep τ used to measure the time between two readings of node remaining energy. A boolean value (*destroy_indicator*) indicating if the node has already been destroyed will also be initialized.

In order to select an appropriate ε value, we relied on some standard energy values for receiving, transmitting, sleeping node activities. We concluded that we have to choose an $\varepsilon \in \left[0.7 \cdot V_{A,\max}, 0.8 \cdot V_{A,\max}\right]$.

The timestep τ has to be chosen based on the compromise between energy consumption in nodes operating procedures and accuracy of the entire process. A timestep τ between 15 and 30 minutes could be a suitable choice.

The second phase (line 04-05 from Fig. 3) is represented by an energy monitoring system, the third phase (line 06 from Fig. 3) by a decisional module that identifies the nodes with low power batteries, and finally the fourth phase (line 07-08 from Fig. 3) by a self-destruction mechanism. These phases are presented in the following paragraphs.

A. *Energy monitoring system*

A specific monitoring system is implemented for every network node to identify if the node's battery reached or will reach in the near future the specified threshold ε.

At every sampling time τ, the monitoring system will read the node battery voltage value $V_A(t)$ by direct requesting information from the network [8] or from a local database [9].

This reading is done using a third-party database in which the specified sensor node writes the battery voltage with the help of an AD converter (Fig. 4).

```
SELECT battery_voltage FROM network_output
      WHERE nodeid = A;
```

Figure 4. Battery voltage reading query

Because in the time interval between two readings the battery can reach the threshold ε, we implemented an autoregressive predictor.

An autoregressive or AR model describes the evolution of a variable only using its past values. This category of systems evolves due to its "memory", generating internal dynamics. Such a model particularized for our node battery voltage variable $V_A(t)$, is defined as follows:

$$V_A(t) = a_1(t) \cdot V_A(t-1) + \cdots + a_n(t) \cdot V_A(t-n) + \xi(t) \quad (1)$$

where $V_A(t)$ is the measurement series under investigation – in our case the node battery voltage, a_i are the autoregression coefficients, n is the order of the auto regression and ξ is assumed to be a Gaussian white noise. Establishing the correct model order n is not a simple task and is influenced by the type of data measurements and by computing limitations of the base station. Reasonable values are between 3 and 6.

After this initialization of the AR-predictor (we will need to collect a couple of $V_A(t)$ readings first), at every instant t we will compute the estimated value $\hat{V}_A(t)$ relying only on past values $V_A(t-1) \ldots V_A(0)$ and we will use both parameter estimation and prediction as follows:

- we will estimate the parameters $a_i(t)$ using a recursive parameter estimation method. From a large number of methods for estimating AR coefficients we decided to use a numerically robust RLS (recursive least square) variant based on orthogonal triangularization, known in

literature as QRD-RLS [9][10]. One of the reasons is that it can be implemented efficiently on the base stations level.

- we will obtain the prediction value $\hat{x}(t)$ using the following equation:

$$\hat{V}_A(t+1) = a_1(t) \cdot V_A(t) + \cdots + a_n(t) \cdot V_A(t-n+1) + \xi(t+1) \quad (2)$$

The corresponding pseudocode for implementing the estimation procedure is presented in Fig. 5:

```
float PREDICT(prior V_A values)
{
  CALCULATE auto regression coefficients a_i;
    // an estimation using QRD-RLS method
  CALCULATE predicted value V̂_A(t);
    // compute sensor predicted value as a result of (2)
  RETURN V̂_A(t);
}
```

Figure 5. Autoregressive prediction pseudocode

B. Decisional module

One of the goals of this methodology is to identify which network node has less energy than needed to stay "alive" in the network. To do this, a decisional module which decides if the current $V_A(t)$ or estimated battery voltage $\hat{V}_A(t+1)$ is lower than a threshold ε has been implemented.

In this case, using a defined TinyOS message and a messaging interface, base station will announce the network coordinator that the investigated node together with its children needs to be destroyed.

Another approach is also available here. We could only destruct the investigated node and then try to perform network self-organization. This way its children nodes will be reached using other routes. However self-organization is an energy consuming task and in some circumstances it must be avoided.

```
typedef nx_struct ExternalActivationMsg
  {
    nx_uint16_t destNodeid;
    nx_uint16_t type;
  }
    ExternalActivationMsg;
```

Figure 6. Activation message

C. Self-destructing procedure

For this step to be efficiently performed, a dedicated software procedure has to be placed at a specified location in each node memory. This code sequence is executed when hosting node receives the self-destruction activation message from network coordinator (it is basically a forwarding process).

The self-destruction procedure consists into several actions:

- Erase node RAM memory that contains susceptible network information, driven software and cryptographic keys and also other additional memories (e.g. flash memory, if present);
- Drain node battery in different ways like R/T radio flood or node logical unit infinite cycle;
- Destroy node radio device;
- Delete node unique identifier from the lists of each of the neighbor nodes, including base station; This way an already captured node won't gain authentication rights if an attacker tries to reintroduce it in the network (the auto-organization property is disabled);
- Mask node measurement nature by hiding the type of the sensor that has been used, of course if node has sensing profile (each node has one or more sensors and knows in a logical way which of them is used for measurements).

The pseudocode for this step is presented below:

```
START thread
{
  CONSUME battery energy
          //broadcast specific messages;
  START thread
  {
    ERASE node memory;
          //erase RAM and flash memory
    DISABLE auto-organization property;
          //delete node identifier from all neighbor's lists
    DESTROY node radio device;
    MASK node measurement nature;
          //for hiding the type of the sensor
  }
}
```

Figure 7. Node self-destruction pseudocode

In the case that we chose to destroy the node together with its children nodes, first we will apply the mentioned self-destruction procedure to all children, and only after that, a self-destruction procedure will be utilized on the investigated node.

IV. CASE STUDY

For validating our methodology, we implemented an experimental network composed four Crossbow MicaZ nodes and a base station (PC/laptop) and has the structure presented in Fig. 8. Every MicaZ node is equipped with one ATmega128L microcontroller, one CC2420 radio device model and three storage units: one program flash (not writeable), one measurement flash and one configuration flash [12].

Also, every node uses 2 AA Panasonic batteries with a nominal voltage of 1.5 V, each. In order to communicate with the base station, network PAN is assisted by a Crossbow MIB520 gateway. All network nodes are running the TinyOS operating system and possess sensing capabilities. The nesC language assures TinyOS programming support [13][14].

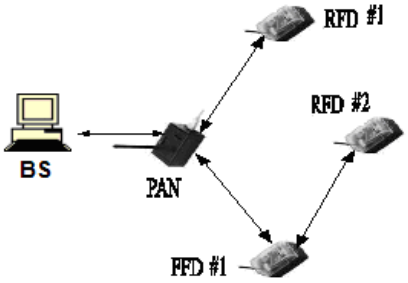

Figure 8. The Crossbow MicaZ network

In our implementation we assumed the threshold $\varepsilon = 2000mV$ and the timestep $\tau = 900$ seconds. The battery discharging evolution obtained for RFD#2 is presented in Fig. 9. In the self-destruction phase (second section in Fig. 9), the battery consumption is done with a higher rate using special configured messages (Fig.10). The reason is that in the self-destruction procedure the node will send useless messages, with large payload data, for rapidly discharging the remaining battery.

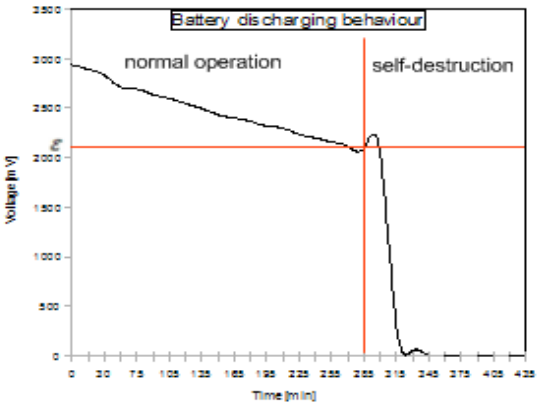

Figure 9. Battery discharging time evolution

```
typedef nx_struct LargePayloadMsg
 {
  nx_uint16_t nodeid;
  nx_uint16_t data[(TOSH_DATA_LENGTH)];
 }
 LargePayloadMsg;
```

Figure 10. Large payload message for battery energy exhaustion

The monitoring system measures and predicts battery discharging behavior, until $\hat{V}_A(t)$ becomes lower than ε. At that moment (approximately after 5 hours of active functioning), the base station sends the self-destruction activation message. The message is forwarded by network coordinator to the specified node, which is starting its own self-destruction procedure.

V. Conclusions

This paper introduces the concept of energy-driven node self-destruction in the cluster-based wireless sensor networks to countermeasure possible threats developed after the operational phase of WSN lifecycle is ended. Based on an efficient power monitoring scheme that discovers the low battery nodes, a self-destruction procedure for that nodes is started.

Acknowledgment

This work was developed in the frame of PNII-IDEI-PCE-ID923-2009 CNCSIS - UEFISCSU grant.

References

[1] Rajaravivarma, V., Yi Yang, Teng Yang, "An overview of Wireless Sensor Network and applications", *Proceedings of the 35th Southeastern Symposium on System Theory*, 16-18 March 2003, pp. 432 – 436.

[2] Becher A., Benenson Z., Dornseif M., "Tampering with motes: Real-world physical attacks on wireless sensor networks", *Proceedings of the 3rd International Conference on Security in Pervasive Computing (SPC2006)*, York, UK, April 2006, pp. 104-118.

[3] Karlof C., Wagner D., "Secure routing in wireless sensor networks: attacks and countermeasures", *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003, pp.113- 127.

[4] Bandara H.M.N.D. and Jayasumana A.P., "An enhanced topdown cluster and cluster tree formation algorithm for wireless sensor networks." *In Proc. of the Second International Conference on Industrial and Information Systems (ICIIS)*, August 2007, pp 37–42.

[5] Ergen S. C., "ZigBee/IEEE 802.15.4 Summary", http://www.eecs.berkeley.edu/csinem/academic/publications/zigbee.pdf, pp. 1-35.

[6] Curiac, D.-l. Banias, O. Dragan, F. Volosencu, C. Dranga, O., "Malicious Node Detection in Wireless Sensor Networks Using an Autoregression Technique", Third International Conference on Networking and Services (ICNS2007), Athens, June 2007.

[7] Curiac D.I., Volosencu C., Doboli A., Dranga O., Bednarz T., Discovery of Malicious Nodes in Wireless Sensor Networks Using Neural Predictors, WSEAS Transactions on Computer Research, Issue 1, Vol. 2, Jan. 2007, pp. 38-44.

[8] Levis Philip, 2006. TinyOS programming, http://csl.stanford.edu/ pal/pubs/*tinyos-programming.pdf*, pp. 1-139.

[9] Crossbow Technology Inc., "Moteworks", Available at http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MoteWorks_OEM_Edition.pdf.

[10] Haller B., Gotze J. and Cavallaro J., "Efficient Implementation of Rotation Operations for High Performance QRD-RLS Filtering", International Conference on Application-Specific Systems, Architectures, and Processors *ASAP 1997, 14-16 July 1997, Zurich, Switzerland*, pp. 162-174.

[11] Apolinrio J.O. Jr, "QRD-RLS Adaptive Filtering", Springer, February 2009.

[12] Crossbow Technology Inc., "Micaz wireless measurement system datasheet" Available at http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf

[13] Gay D., Levis P., von Behren R., Welsh M., Brewer E. and Culler D., "The nesC Language: A Holistic Approach to Networked Embedded Systems", Proceedings of Programming Language Design and Implementation (PLDI2003) San Diego, USA, June 2003.

[14] Gay D., Levis P.. nesC 1.1. Language reference manual. http://nescc.sourceforge.net,2003.