Combined Malicious Node Discovery and Self-Destruction Technique for Wireless Sensor Networks

Daniel-Ioan Curiac, Madalin Plastoi,
Ovidiu Banias, Constantin Volosencu, Roxana Tudoroiu
Automation and Industrial Informatics Department
"Politehnica" University of Timisoara, Romania
daniel.curiac@aut.upt.ro

Alexa Doboli
Department of Electrical and Computer Engineering
University of New York
Stony Broke, New York, USA
adoboli@ece.sunysb.edu

Abstract— **With the continuous development of the wireless technology, securing wireless sensor networks became more and more a crucial but also a demanding task. In this paper we propose a combined strategy that is meant to discover malicious nodes within a wireless sensor network and to expel them from the network using a self-destruction node technique. Basically, we will compare every sensor reading with its estimated value provided by an autoregressive predictor. In case the difference between the two values is bigger then a chosen threshold, the sensor node becomes suspicious and a decision block is activated. If this block decides that the node is malicious, a self-destruction procedure will be started against that specific node.**

Keywords- wireless sensor networks, information security, malicious attacks, self-destruction

I. INTRODUCTION

With the tremendous advances in micro-electro-mechanical systems (MEMS) and radio technologies, a new concept arose - wireless sensor networks (WSN). A wireless sensor network, being a collection of tiny sensor nodes with limited resources (limited coverage, low power, smaller memory sizes and low bandwidth), proves to be a viable solution to many challenging civil and military applications. Their deployment, sometimes in hostile environments, can be dangerously perturbed by any type of sensor failure or, more harmful, by malicious attacks from an opponent.

Sensor networks because of their specific limitations are susceptible to various kinds of attacks that cannot be prevented only by traditional methods (e.g. cryptography): eavesdropping, traffic analysis, selective forwarding, spoofing, wormhole attack, sinkhole attack, Sybil attack and Hello flood attack are the most significant [1]. But, almost certainly the most important danger, due to the inherent unattended characteristic of wireless sensor networks, is represented by node-capturing attack [2], where an enemy acquires full control over sensor nodes through direct physical contact. A node capturing attack is very feasible because of at least two reasons: a) practically, we cannot demand an efficient access control to thousands of nodes distributed in a large territory; and b) it is very difficult to assure tamper-resistance requirements because sensor nodes frequently need to be inexpensive to justify their use.

After an attacker gains the physical control over a sensor node he can extract secret information such as cryptographic keys to achieve unrestricted entrance to higher network levels, or by using reverse engineering techniques he can find security holes to compromise the entire sensor network.

Our proposed countermeasure is based on the fact that a corrupted node is better to be expelled from the network as soon as its malicious activity is started [3]. Even if more sensors are expelled, the WSN will function as designed because of one inherent feature: spatial redundancy [4].

In order to identify a corrupted sensor node, we presumed that even if it may still send authenticated messages (e.g., it can use the cryptographic keys already stored in its memory), it might not operate according to its original specifications sending incorrect readings to the base station. We will identify these sensors by using an autoregression (AR) technique and will eliminate them starting a self-destruction node procedure.

II. SELF-DESTRUCTION METHODOLOGY FOR WIRELESS NETWORK MALICIOUS NODES

In a large number of applications where wireless sensor networks interact with sensitive information or function in hostile unattended environments, it is crucial to develop security related mechanisms.

Due to their nature, these networks have to resist to a plethora of possible attacks. The attackers will try to obtain in-network information or to corrupt the network partially or totally. For making this possible, the attackers will try to gain control over one or several network nodes. Our proposed defending strategy is based on the detection of malicious sensor nodes using AR predictors and the elimination of their effects by expelling them from the network using a self-destruction node technique.

A. Sensor Network Assumptions

In order to assure a high rank of efficiency for our malicious node detection and self-destruction strategy we chose a sensor network having the following features:

- The sensor network is static, i.e., sensor nodes are not mobile; Moreover, each sensor node knows its own position in the field.
- The base station, sometimes named access point, acting as a controller and as a key server, is supposed to be a laptop class device and supplied with long-term power. We also assume that the base station will not be compromised.

- Between the three most common wireless topologies (star, mesh and cluster-tree) we chose a star topology (e.g. Cellular Wireless Network [5] and SENMA [6]) for our sensor network. Star topology is a point-to-point architecture where each sensor node communicates directly with the base station. The main characteristics of the star topology are: there is no node-to-node connections and no multi-hop data transmission; sensor synchronism is unnecessary; sensor do not listen, only transmit and only when polled for; complex protocols are avoided; dependability of individual sensors is much less significant. Because of these features attacks on routing protocols (spoofing, selective forwarding, sinkhole attack, wormhole attack, Sybil attack and Hello flood attack) are almost impossible.
- We rely on efficient secret-key cryptography with pre-distributed keys using Skipjack, RC5 or AES algorithms to encipher all data communications inside the sensor network; These three symmetric encryption algorithms have a common attribute that makes them an attractive alternative in case of sensor networks: they are able to encrypt short or medium size messages, like the ones send by sensors and received by base stations, in the case of limited power consumption. By using such appropriate cryptographic techniques the damaging potential of the passive attacks (eavesdropping and traffic analysis) can be ignored.
- The measurements supplied by each sensor have an important deterministic character, rather than a strictly random (stochastic) one. In this case there exists a correlation between past values and the current one, giving us the power of prediction. As an example, the future values of temperature measurements in a location are strongly related with past and present values, so a prediction method can be applied.

B. Proposed Strategy

Our strategy exploits the temporal redundancy in the sense that previous provided measurements from each sensor will be used to decide if the sensor operates as desired or not. The plan is the following: an attacked sensor node that will attempt to insert false information into the sensor network will be recognized by comparing its output value x with the value $\hat{x}$ predicted using past readings offered by that specific sensor. If the sensor is considered to be malicious, it will be expelled from the network using a self-destruction procedure that will prevent its further undesired activity.

The complete mechanism workflow runs while network is active.

By considering a specific node symbolized by A (in Figure 1), this process is done in the following steps:

a) Associate a node trust indicator with every sensor node. The specified sensor node A will have a trust factor denoted by b_A. This integer value is initially set to zero ($b_A = 0$ for a fully reliable sensor node) and is incremented during our methodology every time a potential malicious activity is encountered. The node trust indicator represents, in other words, the perception of confidence between the WSN and that specific sensor.

b) At every moment in time t, estimate the present value $\hat{x}_A(t)$ provided by sensor node A, using the past readings $x_A(t-i)$ provided by the same sensor A. For sensor A, we can write:

$$\hat{x}_A(t) = f(x_A(t-1),...,x_A(t-n)) \quad (1)$$

where n is the estimator's order. In our approach, an on-line AR predictor performs this step.

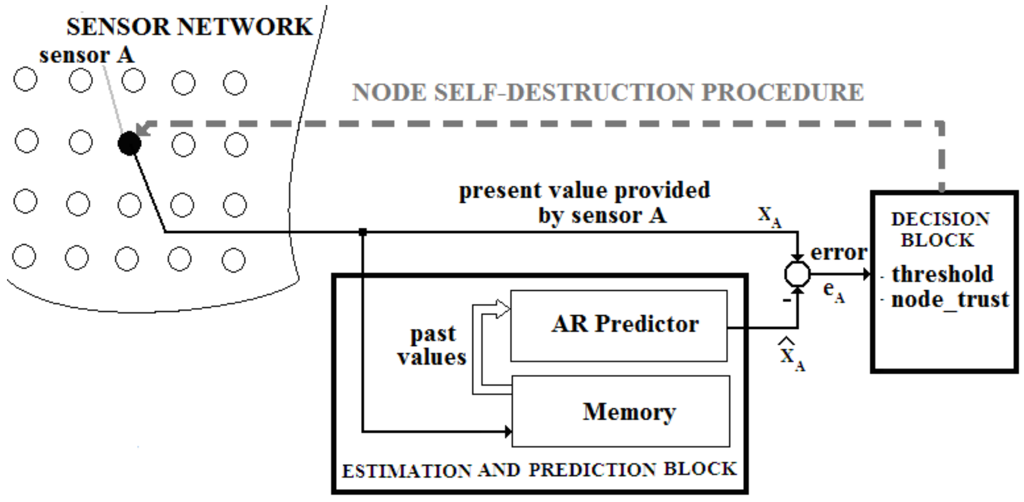

Figure 1. Malicious node discovery and self-destruction

c) Compare the present value $x_A(t)$ measured by the sensor node A with its estimated value $\hat{x}_A(t)$ by calculating the error:

$$e_A(t) = x_A(t) - \hat{x}_A(t) \quad (2)$$

d) Increment the node trust indicator b_A if the error $e_A(t)$ exceeds a given threshold ε : $|e_A(t)| > \varepsilon$ (this is done only one time inside a transitory time zone – due to internal structure of recursive predictors, an error obtained at instant t is propagated/attenuated in the recursive predictor response for some instants, obtaining a transitory regime). If the node trust indicator is higher than a chosen value γ , i.e. $(b_A > \gamma)$, the node will be declared as malicious and a self-destruction procedure will be started for A node.

In Figure 1, we described our strategy, that contains two main phases: a) malicious node discovery, that includes the prediction (done by *Estimation and Prediction Block*), the comparison and the decision making (done by *Decision Block*); and b) node self-destruction procedure.

The pseudocode is presented in Figure 2:

```
FOR (every sensor node)
 {
  …
  SET $b_A$, $\varepsilon_A$, k;  //node trust indicator, threshold and the
    //approximation of length of predictor's transitory regime
  }
WHILE (network is active)
{
 …
 FOR (every sensor node)
 { $x_A$ =READ sensor A; //get sensor actual value
  …
  $\hat{x}_A$ =PREDICT(prior $x_A$ values); //call prediction method
  $e_A$ = $x_A$ - $\hat{x}_A$; //calculate the error
 IF (ABS ( $e_A$ ) is greater than $\varepsilon_A$ )
  IF (predictor is not in transitory regime)
   {
    $b_A$ = $b_A$ + 1; //increment node trust indicator
     START thread TRANSITORY_REGIME;
            //a counter set on k and will be decremented
            //every instant until it becomes zero
    DECISION_BLOCK (sensor A);  //call decision method
  }}
  …
}
```

Figure 2. Strategy implementation pseudocode

III. Malicious Node discovery

Our stratagem to identify a corrupted sensor node is based on the fact that, even if it may still send authenticated data, it may not operate according to initial requirements sending incorrect readings to the base station. These nodes will be identified in the moment they begin to broadcast erroneous information by using a linear autoregressive predictor.

In order to implement this predictor we considered that an autoregressive (AR) model efficiently approximates the evolution in time of the measurements provided by each sensor. An autoregressive or AR model describes the evolution of a variable only using its past values. This class of systems evolves due to its "memory", generating internal dynamics, and is defined as follows:

$$x(t) = a_1 \cdot x(t-1) + ... + a_n \cdot x(t-n) + \xi(t) \qquad (3)$$

where x(t) is the measurement series under investigation, a_i are the auto regression coefficients, n is the order of the auto regression and ξ is assumed to be a Gaussian white noise. By convention the time series x(t) is assumed to be zero mean. If not, another term (a_0) is added in the right member of equation (3). 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.

If the a_i coefficients are time-varying, the equation (3) can be rewritten as:

$$x(t) = a_1(t) \cdot x(t-1) + ... + a_n(t) \cdot x(t-n) + \xi(t). \quad (4)$$

The model (4) can be used either to estimate the coefficients $a_i(t)$ in case the time series x(t),…, x(t-n) is known (recursive parameter estimation), either to predict future value in case that $a_i(t)$ coefficients and past values x(t-1),…,x(t-n) are known (AR prediction).

A. Autoregressive prediction

In the beginning we have to associate a threshold $\varepsilon > 0$ with every sensor node. This threshold will be used to decide if a sensor operates normal or abnormal and its value depends on the type of the sensor and its specific functioning in-field conditions. For a specific sensor A, the threshold will be denoted by ε_A .

After this initialization, at every instant t we will compute the estimated value $\hat{x}_A(t)$ relying only on past values x_A(t-1),…, x_A(0) and we will use both parameter estimation and prediction as in the following steps:

First step: 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 [7]. One of the reasons is that it can be implemented efficiently on the base stations level (laptop class device).

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

$$\hat{x}_A(t) = a_1(t) \cdot x_A(t-1) + ... + a_n(t) \cdot x_A(t-n) + \xi(t). \quad (5)$$

The corresponding pseudocode for implementing the estimation procedure is presented in Figure 3:

```
float PREDICT(prior $x_A$ values)
{
  CALCULATE auto regression coefficients $a_i$ ;
        // an estimation using QRD-RLS method
  CALCULATE predicted value $\hat{x}_A$ ;
        // compute sensor predicted value as a result of (5)
  RETURN $\hat{x}_A$ ;
}
```

Figure 3. Autoregressive prediction pseudocode

After that, we will compare the present value $x_A(t)$ measured by the sensor node with its estimated value $\hat{x}_A(t)$ by computing the error using (2).

B. *The Decision Block*

If the error computed by (2) is higher than the threshold ε_A, then the sensor A will be considered to be a potentially corrupted sensor and the *Decision Block* will be activated (Figure 1).

Our implemented *Decision Block* is based on the counter b_A , which is incremented every time that $|e_A(t)| > \varepsilon_A$. If b_A exceeds the limit γ, the sensor is declared to be corrupted and the self-destruction procedure is started for that node. The corresponding pseudocode is presented in Figure 4:

```
void DECISION_BLOCK (sensor A)
{
 IF ( $b_A$ greater or equal than $\gamma$ ) //sensor is corrupted
  {
   SELF_DESTRUCTION (sensor A) ;
           //call self-destruction mechanism for sensor A
  }
}
```

Figure 4. Decision block pseudocode

The precision of this decision block can be increased using a concurrent technique based on the values provided by neighboring sensors such as [8]. In this case, the decision taken on behalf of two criteria to eliminate the attacked sensors will be more accurate.

IV. Malicious Node Self-Destruction

If a certain node has been tagged as malicious, base station will initiate a self-destruction sequence for that specific node. The self-destruction routine is divided 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 [9];
- 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 (will disable auto-organization property);
- Mask node measurement nature by hiding the type of the sensor that has been used (each node has one or more sensors and knows in a logical way which of them is used for measurements).

The above actions have to be performed in order of their importance, although some kind of concurrency could be assured. For example: the initiation of self-destruction could start the procedure for draining node battery, but in the same time it could conduct erasing actions for memory and cryptographic keys.

Self-destruction should take into consideration all network characteristics from design to deployment including the topology. Also it strongly depends on the node hardware profile. For the proposed star network model, self-destruction will imply only the base station and the compromised node – as we stated earlier, each node communicates directly with the base station.

Basically, self-destructive sequence may be a software routine embedded into node's memory or sent bit by bit from the base station to the aimed node. Entire code has to be compatible with nodes and base station operating system. The pseudocode for node self-destruction is presented in Figure 5.

In the optimistic scenario, after self-destructive routine was initiated, the intended node is destroyed and obvious, undetectable in the network. All references to its identifier will disappear from all network devices. Cryptographic keys and stored information will be also deleted.

In the pessimistic scenario, the self-destructive response won't have any triggered action attached to it; the corrupted node will still be alive in a fully or partially functional state. This scenario will have to be avoided by reducing the probability of some unfortunately events like:

- Self-destructive routine was not suitable implemented for node hardware profile;
- The node's software is modified by the enemy in such way that it doesn't accept incoming messages from the base station, it only sends malicious data;
- Node battery is almost exhausted at the moment of executing the routine. In this case no memory erasing will be performed. The attacker will replace the batteries and the node will be partially or totally running.

```
void SELF_DESTRUCTION (sensor A)
{
 WHILE (sensor A is in network)
 {
  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 5. Node self-destruction pseudocode

Our solutions for avoiding these incidents are: testing of the self-destruction code on every type of sensor nodes that is included in the WSN; hiding the self-destruction routine into the node's memory; paying attention to the energy consumption on each sensor.

V. CASE STUDY

For validating our strategy, we used a CrossBow wireless sensor networks, containing 15 Mica2 nodes and one base station in a star topology. The sensors were set to measure the temperature in a room. In order to model an attack upon sensor A we made the following experiment: at specific moments in time we intervened with an electric incandescent lamp (200 watts) placed very near to sensor A, for disrupting the normal functioning of this specific node (Figure 6).

We chose the predictor's order $n = 3$, the threshold $\varepsilon_A = 1^\circ C$ and the limit $\gamma = 3$. Also, we approximated the time length of the predictor's transitory regime at $k = 5$ instants.

In Figure 7a we represented the sensor's A output time series, including our two "malicious" interventions at instants t=15sec, t=20 sec and t=27 sec. In Figure 7b we presented the time variation of the measured and predicted time series, and in Figure 7c we presented the error $e_A(t)$.

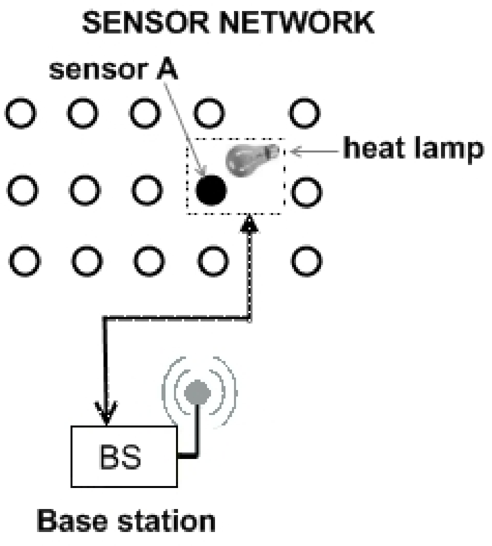

Figure 6. Case study description

The results are as expected: after exceeding the threshold ε_A for three times, the sensor A is expelled from the WSN by starting the self-destruction procedure. This result can be observed in Figure 7 where no more readings are obtained from sensor A after t=27 sec.

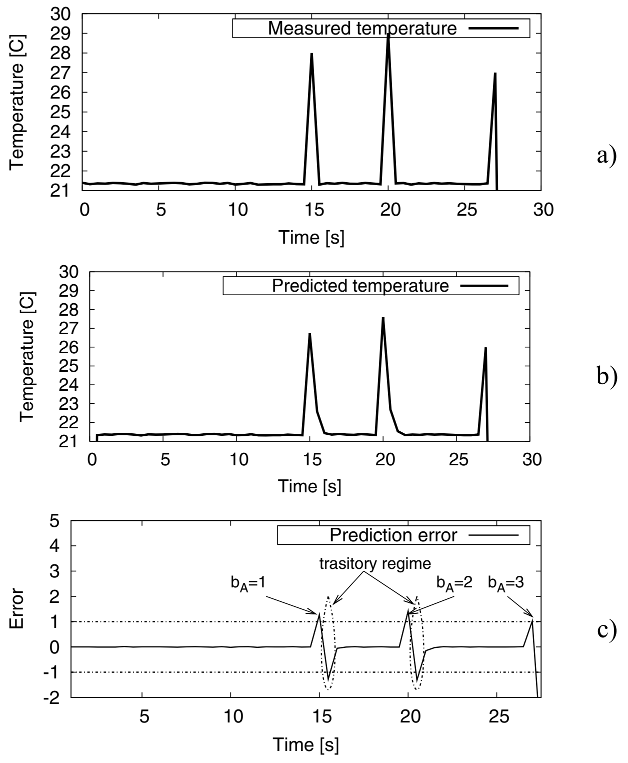

Figure 7. a) The sensor's output time series; b) Predicted time series; and c) prediction error time series

VI. Conclusions

Security issues related to WSN become more and more an important research area. Detecting abnormal/malicious function of motes and offering efficient countermeasures represents a difficult task. In this paper we propose a combined strategy that not only detects the corrupted nodes, but also excludes their malicious activity using a self-destruction node technique. Due to the inherent spatial redundancy feature of WSN, applying a self-destruction procedure to corrupted nodes has no major inconveniences, extending the secure function of the entire sensor network.

Acknowledgment

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

References

[1] Karlof C. and Wagner D., “Secure routing in wireless sensor networks: attacks and countermeasures”, Proceedings of the 1st IEEE International Workshop SNPA2003, Anchorage, USA, May 2003, pp. 113-127.

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

[3] D.I. Curiac, O. Banias, F. Dragan, C. Volosencu, O. Dranga, “Malicious Node Detection in Wireless Sensor Networks Using an Autoregression Technique”, Proceedings of the 3rd international conference on networking and services, ICNS2007, Athens, Greece, June 2007.

[4] Y. Gao, K. Wu, and Fulu Li, “Analysis on the Redundancy of Wireless Sensor Networks,” Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, ACM WSNA’03, San Diego, USA, September 2003, pp.108-114.

[5] J. Feng, F. Koushanfar, and M. Potkonjak, “System-Architectures for Sensor Networks Issues, Alternatives, and Directions”, Proceedings of International Conference of Computer Design, ICCD’02, Freiburg, Germany, Sept. 2002, pp.226-231.

[6] L. Tong, Q. Zhao, and S. Adireddy, “Sensor Networks with Mobile Agents", Proceedings IEEE 2003 MILCOM, Boston, USA, October 2003, pp.688-694.

[7] B. Haller, J. Gotze, and J. Cavallaro, “Efficient Implementation of Rotation Operations for High Performance QRD-RLS Filtering”, ASAP '97 Proc., 14-16 July 1997, Zurich, Switzerland, pp. 162-174.

[8] D.I. Curiac, C. Volosencu, A. Doboli, O. Dranga, and T. Bednarz, “Discovery of Malicious Nodes in Wireless Sensor Networks using Neural Predictors”, WSEAS Transactions on Computer Research, Issue 1, Volume 2, January 2007, pp. 38-43.

[9] A. A. Pirzada, and C. McDonald: “Kerberos assisted Authentication in Mobile Ad-hoc Networks”, Proceedings of the 27th Australasian conference on Computer science - Volume 26, 2004, pp. 41-46.