A lightweight threat detection system for industrial wireless sensor networks

Y.B. Saied, A. Olivereau

A Lightweight Threat Detection System for Industrial Wireless Sensor Networks

Yosra Ben Saied[1], Alexis Olivereau[2]
(Corresponding author: Yosra Ben Saied)

RAMSIS Group, CRISTAL Lab, National School of Computer Science, Tunisia[1]
CEA, LIST, Communicating Systems Laboratory, Gif-sur-Yvette, France[2]
(Email: yosra_bensaied@yahoo.fr)
(Received Aug. 13, 2015; revised and accepted Sept. 29, 2015)

Abstract

Wireless Sensor Networks (WSNs) led the way to new forms of communications, which extend today the Internet paradigm to unforeseen boundaries such as eHealth, intelligent building or smart grid, to name a few. The legacy industry, however, is slower to adopt this technology, mainly for security reasons. Self-managed security systems allowing a quicker detection and better resilience to attacks, may counterbalance this reluctance. In this article, we propose a hybrid threat detection system that involves both centralized decision and local, per-cluster, work repartition and that is designed to run on top of industrial wireless sensor networks. Compared with the literature, we prove that this system is suitable for architectures mainly composed of constrained and sleeping devices, for which it achieves a fair level of autonomous security without prohibitively drawing out energy resources.
Keywords: Energy efficiency, hybrid approach, intrusion detection, resource constraints, wireless sensor network

1 Introduction

Wireless Sensor Networks (WSNs) changed the Internet communication paradigm by introducing unattended devices in a hitherto human-centric model. WSNs filled the gap between the physical world and the Internet by offering the ability to monitor in real time a wide range of physical values through connected devices.

Initially built around very simple entities, technologies and architectures, WSNs benefitted from a strong interest from the international research community, which designed in a few years a wide variety of new components and systems. Dedicated radio technologies and routing protocols were specified, as well as novel, complex architectures. These architectures evolved from a basic, unidirectional topology made of multiple sensor nodes pushing data to a server through a sink node, to more advanced systems featuring bi-directionality. These new systems paved the way towards the emerging Machine to Machine (M2M) and Internet of Things (IoT) frameworks.

This gain of maturity of WSN technologies accelerates their adoption in the industry, and this adoption is all the quicker as WSNs answer to classical needs of industrial scenarios: monitoring of physical values, asset supervision and facilities surveillance are all key requirements in these scenarios, for which dedicated sensor nodes are available. Introducing actuators along with sensor nodes allows for more complex operations such as asset control and better production chain automation. Finally, the addition of more complex entities, such as indoor positioning nodes and mobile devices, to an existing sensor topology can give rise to new services, also profitable for the industry. For example, better worker safety can be achieved if the production chain is able to detect a worker's presence in an unauthorized area, or to determine that a certain user is not wearing adequate protection suit.

However, even though cost effective devices and energy-efficient technologies and protocols are available, the underlying security question impedes the use of WSNs in the most critical industrial scenarios. The inherent vulnerability of WSN nodes, which is due to their exposed location and their use of wireless communications, is such that a WSN has to mimic all security features from the legacy Internet, while also adding specific use cases and taking into account the strong shortcomings of the WSN nodes. Yet, strong security is often contradictory with limited resources. Furthermore, the unattended nature of WSNs and their autonomous decision taking upon a context change put them at risk of triggering harmful behavior, if misled by an attacker.

We propose to leverage on this WSN autonomy to provide it with the ability to detect new threats and react to them in the most appropriate manner. Contrary to existing work, the proposed threat detection system is lightweight enough to be run on resource-constrained sensor nodes. It greatly improves the resilience of the industrial WSN, without bringing in excessive energy con-

sumption. Our solution is based on a partly centralized architecture and specifies new roles for WSN entities, in accordance with their status and capabilities.

The remainder of this paper is organized as follows. Section 2 quickly describes challenges related to the development of a threat detection framework for WSNs and highlights the limitations of existing approaches. Section 3 describes the proposed threat detection solution and its implementation on physical sensor devices. We assess its performance when compared to main state of the art solutions in Section 4. Section 5 concludes the paper.

2 Problem Statement and Related Work

The need for a threat detection system arises from the need to protect WSNs against attacks. Obviously, we assume that WSNs feature dedicated authentication and access control routines, key establishment functions. The security here is provided by the use of cryptographic primitives, which protects the WSN from external attacks. However, these security functions fail to protect the network against an authenticated attacker from inside the network: a sensor node owning legitimate cryptographic keys can easily launch an internal attack inside the group. It may be reluctant to make its resources available to other nodes as part of a cooperative act. Therefore, such a node may prefer to behave at times in a selfish manner and refuse to cooperate in packet forwarding in order to maximize its energy savings. On the other hand, a legitimate sensor node can also act maliciously by dropping packets, delaying the transmission or sending packets through a different route than planned. Such an internal attacker can only be detected through behavioral analysis mechanisms. The latter track the system behavior and interactions between nodes to detect threat attempts and/or occurrences. Once a security anomaly is detected, a reaction mechanism is launched to take security and service repair measures.

While highly relevant to the autonomous topologies that wireless sensor networking involves, threat detection raises issues with respect to its adaptability to these domains. It challenges indeed the constrained nodes' limited energy resources by involving complex processing and high resource consumption.

Most of existing researches on threat detection systems propose to mimic models proposed for Mobile Ad hoc NETworks (MANETs) in order to counter internal attacks inside WSNs [4, 8, 9]. These threat detection models rely on monitoring agents deployed on each sensor node. The goal of these local agents is to track the traffic within their radio range and detect misbehaving nodes causing routing disruptions. Authors in [10] define a threat detection system based on monitoring agents. They distinguish between local agents (able to process only the packet they actively forward) and global agents. Global agents act as watchdogs by monitoring nearby traffic, especially when they determine how a forwarding node has behaved (e.g. difference, delay and loss between an incoming packet at a neighbor). In addition to regularly monitoring global agents, the authors introduce the concept of spontaneous watchdog behavior: a node determines from a sensed packet that it can be in position to act as a watchdog for this packet. This node will first identify the number n of its neighbors that could also act as watchdogs for the same packet, and turns on its watchdog behavior for that packet with a one-in-n probability. However, local passive monitoring has been shown to draw as much power from the sensor node transceiver as data reception [13]. Ioannis et al. in [7] also apply the watchdog behavior, with a higher emphasis on countering malicious monitoring nodes. To that aim, cooperation through majority is achieved, relying on an encrypted flag. That means that a node will be classified as malicious only if a majority of its neighbors flag it as such. The problem with the use of watchdog behavior is that watchdog nodes will drain their resources quickly.

Butun et al. in [3] review existing intrusion detection systems in WSNs and highlight the fact that IDSs designed for MANETs cannot be applied to WSNs directly. Authors consider that resource constraints characterizing the sensor nodes should be taken into account for the design of an IDS adapted to WSNs. Also, the fact that most of sensor nodes are most of the time in sleeping mode makes the operation of this security system more complex, because synchronous node actions might prove difficult, if not impossible, to achieve.

Other researches recognize that designed IDSs need to spend the least amount of energy as possible to spare enough energy for the crucial operations of the WSN. For this reason, Huang and Lee propose in [6] an energy-efficient monitoring system that fit the energy constraints of sensor nodes. They select a single node to perform monitoring at a given time within a given cluster. This node is designated through a fair election process. Authors claim that they obtain much higher energy efficiency at an equivalent security level compared with systems where all nodes perform monitoring at the same time. They also provide a detailed set of rules that allow detecting classical attacks expected to occur within a wireless sensor network. Authors in [1, 12, 14] propose hierarchical trust management systems for WSNs to detect selfish and malicious nodes. In these approaches, regular sensor nodes do not participate in the global decision making process. Only Cluster Heads (CHs) are responsible for the global decision making process and the response. The main reason for this is to reduce the energy consumption. They wanted to conserve the energy of the majority of sensor nodes, by simply assigning them as subordinates under CHs.

These cluster based detection schemes consider that a node is periodically elected to be the monitoring agent within each cluster. However, the election process could be heavy for constrained nodes -as demonstrated in what

follows. In the same context, Butun et al. in [3] underline that clustering based IDSs may consume considerable amount of the WSN's energy through exchanged messages between nodes to periodically elect new monitoring agents.

3 Solution Description

3.1 Overview

Considering the inadequacies of existing approaches described in the previous section, we propose an efficient threat detection system for WSNs that aims at meeting two key requirements. Considering the energy constraints of sensor nodes will be the first key requirement for the design of our system. Processing within sensor nodes will be minimized in order to increase their lifetime. So that, most of required operations are delegated to the server side and the charge on the sensor node will be kept light, which leads to extend the network lifetime. Offloading the charge from the constrained nodes to the server is not the only advantage of this approach. Having to send its observations to the central server to take decisions, a malicious node would not be in position to propagate false assessments about specific victims. With a central entity responsible for decision making, it becomes a common profit for all nodes to provide reliable evidences since false ones can globally affect decision making at the central entity, and could eventually be detrimental to the attacker itself.

The second important key requirement is that a sensor node is often in a sleeping mode, so that considering that it is able to perform synchronous actions with other nodes for threat detection as proposed in previous works would be difficult to achieve. This problem will be carefully taken into account for the design of our solution. Sensor nodes will wake up only to collect data and perform autonomous security-related actions, then revert back to sleep mode.

3.2 Network Model

The wireless sensor network is supposed to be divided into zones. Each zone contains one or more clusters and each cluster contains one or more sensor nodes. Zones and clusters have different criticalities, different security levels and different security policies. It is also assumed that sensor nodes within the same cluster can communicate with each other. In addition, we assume that the awake time is negligible compared to the sleep time. Finally, we exclude any form of synchronization between nodes.

In a WSN, there are two types of nodes: sensing devices and gateways (or servers). The sensing devices are simple nodes equipped with radio interfaces only. The gateways are equipped with an Ethernet port to communicate with remote application servers and deliver collected data. The Gateway has more computing capacities and unlimited energy resources, as compared with a constrained sensor node. Since these nodes are Ethernet-connected, assuming that they are also connected to a power supply seems a reasonable hypothesis.

3.3 Components and Roles

The security system we present in this paper is made up of the following elements:

- *Threat Detection Client*: the TD client is in charge of identifying threats and sending reports to the TD server.
- *Threat Detection Server*: the TD server receives the registration requests from sensor nodes. It chooses which sensor(s) will be in monitoring mode for each cluster by taking into account status parameters such as batteries level and available resources. The TD server updates the global network database. It receives the alarms from TD clients and transfers them to the SA Server.
- *Security Adaptation Server*: the SA Server receives the threats from the TD server. It decides then which is the best policy to apply accordingly and stores the new policy for sensor nodes in the security policy mailbox.
- *Security Adaptation Client*: After each boot, the SA Client sends a message to the SA Server to check if there are any policies to be delivered and applies those it receives in return.
- *Security Service*: Various security services are supposed to run on the considered sensor nodes, but two of them are especially relevant. A key management service is assumed to be able to establish security contexts between two nodes. The security level offered by this service translates into various parameters such as algorithms, key lengths or security association life-times. Another security service is the network access control enforcement, which maintains a secure connection between a sensor node and its parent node in the WSN topology. This secure connection is established upon node's entry into the WSN, and remains active as long as the node is part of the WSN. Both of these security services can raise alarms to the local threat detection client even though the node is not actively monitoring its neighborhood, for example if a peer node cannot be authenticated (by other sensor in case of key management, by parent node in case of network access control enforcement).
- *Security Policy mailbox*: a module that stores the generated policies. Then, it delivers them at nodes request. The use of this module is required since the nodes are not synchronized. It also reduces the overall bandwidth consumption.
- *Global Network Database*: contains a global view of the network and the threats detected in the past.

It is populated every time a sensor registers to the TD Server and is updated every time a new policy is applied by the SA Server.

3.4 Operation

This section describes the sequences of actions performed by the client-side and server-side entities. These sequences of actions are categorized according to the sensor mode. This mode can either be bootstrapping, normal mode, or monitoring mode.

Bootstrapping is the mode of a node when it joins the network. The newly joining node is first to send a registration request to the TD server, informing this latter about its potential monitoring abilities. Upon reception of this registration request, the TD server registers the node and sends to the node an acknowledgement (ACK), as well as a configuration message specifying whether the node should remain in normal mode or switch to monitoring mode for a specific amount of time. The decision by the TD server is based on its knowledge of current and, in some cases, foreseen contexts of the candidate monitoring nodes. This contextual information includes data relevant to the nodes resources (e.g. battery level), location, and capabilities (e.g. number of observable neighbors). With this information, the TD server is able to identify the best node in the cluster for acting as a monitoring entity, and to configure it with this role for a certain period of time. Once the monitoring delay expires, the TD server proceeds again to the identification and designation of cluster monitoring node(s).

A node switches to **Monitoring Mode** when ordered to do so by the TD server, either immediately after its bootstrapping, or upon reception of a configuration order, after waking up. The sequence of actions performed when in monitoring node is:

1) Once the TD Client detects a threat, it sends an alarm to the TD server that includes information about the threat. This information contains at least the IP address of the attacker, the IP address(es) of the target(s) and the type of attack;

2) Upon receiving the alarm, the TD Server reports it to the SA server, optionally after having aggregated multiple alarm messages and/or having assessed the quality of the evaluator. Next, it stores the new policies in the security policy mailbox, in order to have them be delivered to the respective SA Clients that will have to enforce them;

3) In monitoring mode, the TD client on the sensor regularly polls the security policy mailbox by sending a dedicated inquiry message to the SA server;

4) The SA server sends the requested policy if it exists. Otherwise, it replies with a message telling the node that it is not to enforce a new policy. Along with security policies, the monitoring node might be instructed to revert back to normal mode if another monitoring node is being designated within the cluster;

5) If a new policy is received, the SA Client enforces it by configuring the security services in accordance with the received policy and acknowledges it through an ACK message;

6) The SA Server receives the ACK and updates the global network database accordingly.

The monitoring mode sequence of actions is represented in Figure 1.

If a malicious node refuses to be placed as a monitoring node, the server could detect that no reports are received from its side and then it could be penalized or excluded from the network. If this node accepts the monitoring process during a period of time but lets other colluding nodes execute attacks while sending positive observations on their behalf. The server can compare its false reports with feedbacks received from other monitoring nodes selected for the same cluster in different time intervals. Hence, it will conclude its malicious behavior as a monitoring node and take the appropriate punishment decision against it.

The **Normal Mode** is the default mode for a bootstrapped sensor node that has not been designated as a monitoring node. In this mode, sensor nodes alternate between active and sleeping states. Upon leaving sleeping state, the node interrogates the SA server about an eventual new policy to enforce. Meanwhile, the node in normal mode may also be instructed to switch into monitoring mode, if required from the evaluation of different nodes energy levels within the considered cluster. The node then performs the task(s) for which it has left the sleeping state. An alarm may be raised by a node in Normal Mode only if one of the run tasks detects a threat. This task would then notify the TD Client through an API call.

The sequence of actions corresponding to normal mode is represented in Figure 2.

3.5 Implementation Environment

The hybrid threat detection solution presented in this paper was implemented using embedded C for sensor nodes, with Atmel studio 6 for AVR/ARM and C++ for the server side. We used Dresden Elektronik sensor devices [5] that have the following specification:

- The gateways are equipped with ARM processor, Ethernet port, power over Ethernet, USB/Serial and IEEE 802.15.4 2.4GHz.

- The sensor devices are equipped with AVR Processor, battery, USB/Serial interface and IEEE 802.15.4 2.4GHz.

Both the gateway and the sensor node use 6LOwPAN protocols. The sensor node turns to the Monitor Mode

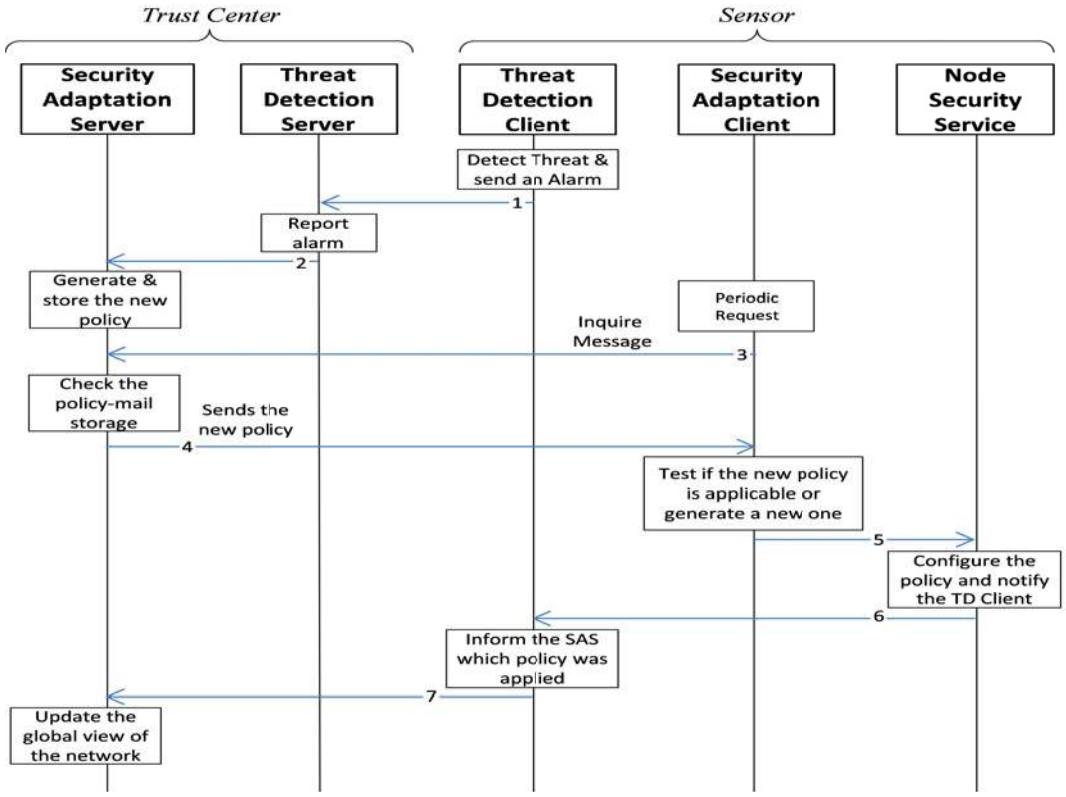

Figure 1: Threat detection and security adaptation in monitoring mode

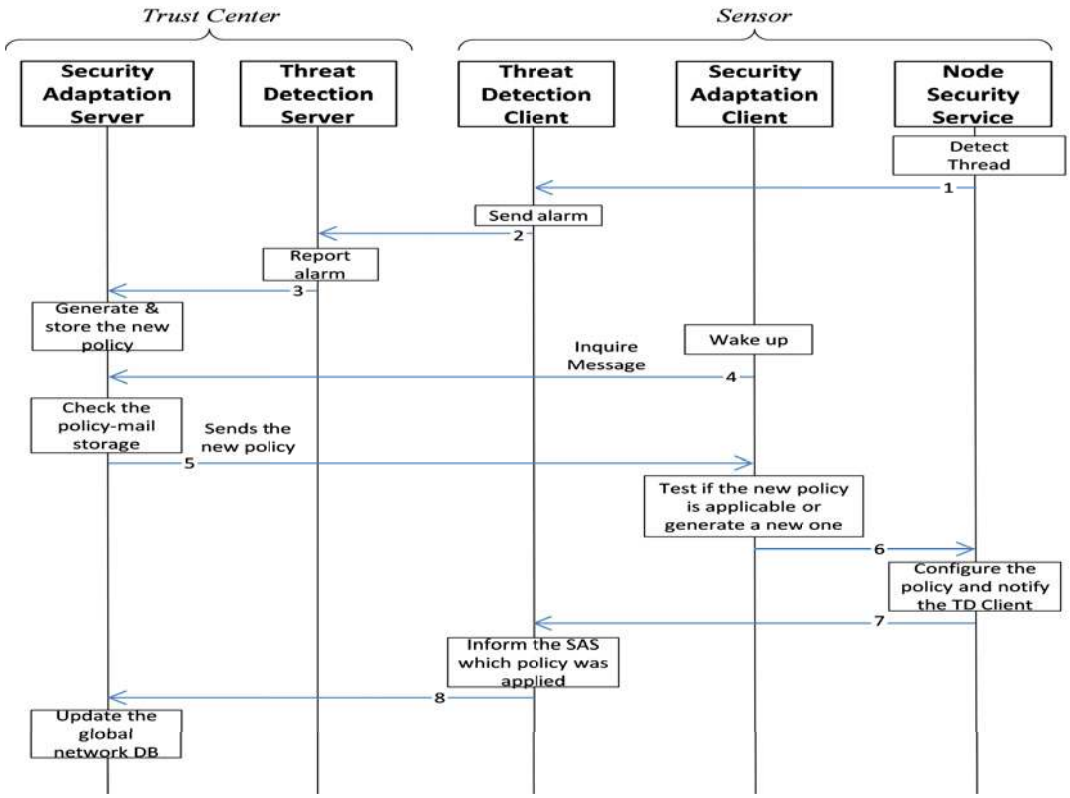

Figure 2: Threat detection and security adaptation in normal mode

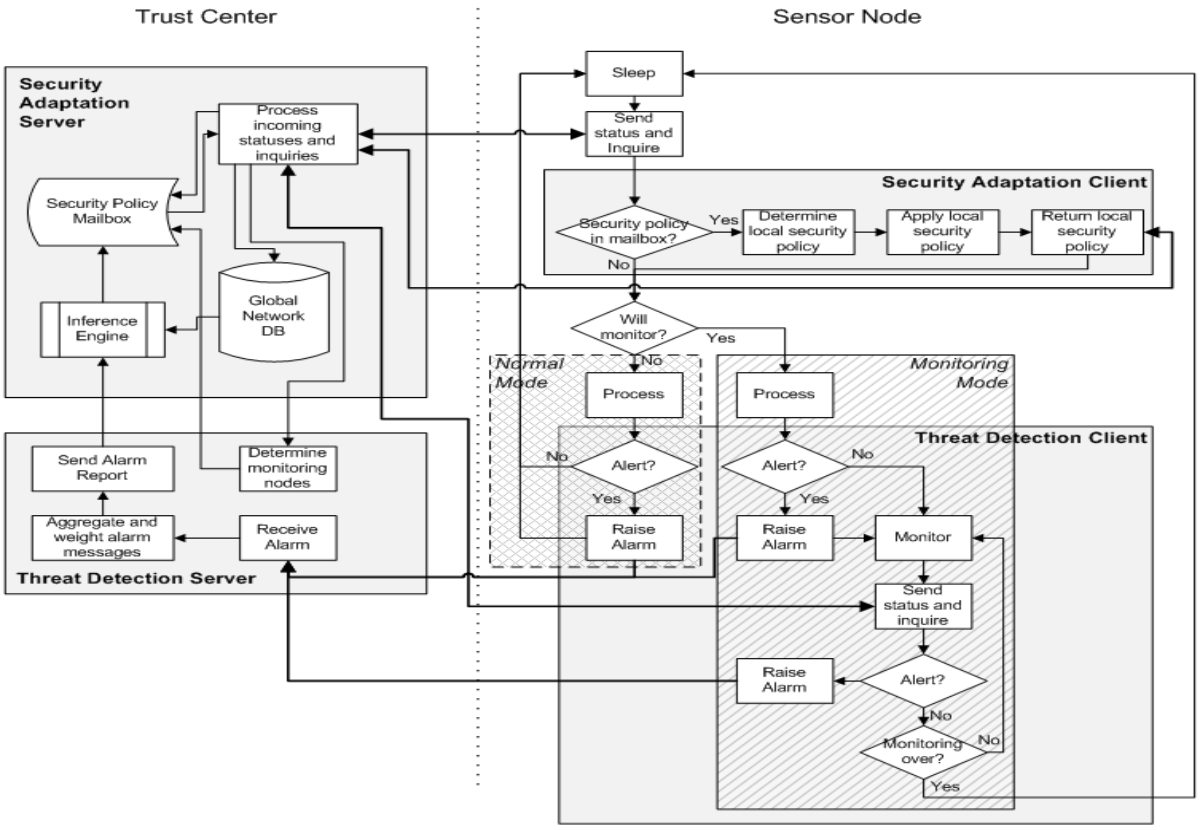

Figure 3: Overall logical architecture and state machines

by switching its 802.15.4 interface to promiscuous mode and disabling its sleep behavior. The overall process of our solution, illustrating its state machines and internal/network message exchanges, is depicted in Figure 3.

4 Performance Evaluation

4.1 Performance Evaluation Criteria

In order to assess the performance of the proposed hybrid threat detection system, it is worth detailing first all criteria that should be taken into account for the evaluation of a threat detection system. Indeed, a threat detection system can be characterized with:

1) The *proportion of false positives* (benevolent behaviors wrongly identified as attacks) and that of false negatives (malicious behaviors not identified as such), as compared with the successful attacks identifications. These ratios and the corresponding performance metrics assume the existence of a list of detectable attacks $\{A_1, \ldots, A_n\}$. Accordingly, an attack A' not detectable - and thus not detected - by the system would not be qualified as a false negative.

2) The *exhaustivity of the detectable attacks list* constitutes a second evaluation metric. Ideally, this list must be built in accordance with a risk analysis carried out on the monitored system, so that the most frequent and/or critical attacks be detectable by the threat detection system.

3) The *reactivity of the threat detection system* that is, the time needed to identify an attacker as malicious. Finally, constrained environments characterized with nodes of low capabilities in terms of computing power and energy capacity exhibit additional requirements, to which a fourth evaluation criterion corresponds.

4) The *energy needed* to locally (at constrained nodes side) operate the threat detection system. This energy cost must be limited, so that the constant use of this system will not consume a high amount of nodes energy re-sources prohibitively fast.

The proposed solution for threat detection does not focus on mitigating false positives and false negatives. From this viewpoint, the local detection algorithm is not improved as compared to the state of the art solutions. Likewise, the weighting and aggregation of alerts at the server side is recommended, but a novel method for doing so is not part of the current proposal. Hence, our proposed system will not be evaluated according to the first evaluation criteria. Our proposed system does also not extend the list of detectable threats, as defined by the second evaluation criteria. The completeness of this list is seen

as orthogonal to the current solution. We acknowledge, however, the high relevance of this problem, especially when put aside the (often neglected) nodes constraints in terms of memory capacity. Indeed, an exhaustive list of detectable attacks would require a corresponding memory space for storing attacks signatures, which would not be affordable to highly-constrained nodes.

While criteria (a) and (b) are thus not considered for assessing the performance of our proposed threat detection system, the importance we put in the reduction of energy consumption leads us to favoring criteria (d) as the most important one for doing so. Meanwhile, we will use the criteria (c) both to guarantee a proper behavior of the proposed threat detection system even though it is instantiated on highly-constrained nodes and to provide a second metric to compare our solution to those proposed in the literature.

4.2 Threat Detection Simulator

We evaluated the performance of our proposed threat detection system by means of a purposely designed discrete time simulator. This section first reviews the generic design decisions that were applied for conceiving this tool. It then considers the models that were used to implement on the simulator both the proposed system and the solution proposed in [6], which is one of the most studied threat detection system for WSNs, to which we intend to compare our work.

The proposed simulation program Csimu (a lightweight discrete time C-based network simulator) starts with initializing a WSN-like topology made of clusters and cluster heads. It also initializes the threat detection system, in particular by assigning per-cluster initial monitoring nodes. We conceived our simulator as a program built around a main loop, with each loop execution corresponding to an elapsed (configurable) time step. A discrete time approach is therefore adopted in this tool. The main loop is made of the following operations:

- Storage of logged values including: overall energy spent, overall energy spent for all threat detection related operations, nodes compromise status.
- Per-node individual processing loop. This second loop is the most important operation. It goes through each node part of the generated topology and performs relevant nodes actions. These consist in:
- Storing per-node values relative to energy spent and energy spent for threat detection operations.
- Updating node energy level in accordance with node status during the last time step. For example, a node that spent the last time step in 'Listen' mode will have its battery level decreased with an energy $E = \Delta t_{Timestep} \times P_{Listen}$, with $\Delta t_{Timestep}$ being a configurable parameter and P_{Listen} the power for the considered node type in listening mode.
- Carrying out node tasks, e.g. uploading a measured value to a remote sink node and updating node energy level accordingly.
- Carrying out attacks, in case the node is compromised. Two types of attacks are represented: A_{DoS} is an example of Denial of Service attack and A_{Comp} represents a compromise attack. Each attack involves the following steps: determination of whether an attack is to occur, choice of the to-be-attacked victim, energy consequences on the attacking node (corresponding to the number of messages to send/receive and the listening durations involved by the attack), energy consequences on the victim, other outcomes on the victim (the A_{DoS} attack has a chance to make the victim compromised), attack detection. The attack detection involves all nodes that are in position to detect the launched attacks, taking in considerations their respective locations (only neighbor nodes are susceptible to discover the attack) and modes (the probability of detecting the attack is much higher for the nodes that are in monitoring mode). In turn, when a node detects an attack, it may either be able to pinpoint the attacker, or may just notice that an attacker is likely to be present in the cluster. In accordance with the detection results and the implemented threat detection system, various actions are to be taken that are detailed in the next section.
- Threat detection system update (detailed in the next subsection) that includes the exclusion of the identified compromised nodes and, in random (non-scripted) mode, a slight probability for each node to become compromised.

4.3 Evaluated Threat Detection Systems

4.3.1 "All Monitor" and "Cluster Monitoring Node Election" Systems

As explained in the related work section, the "all monitor" approach considers that each sensor device acts as an independent monitoring node. Continuing in listening mode, each node tracks its neighbor's communication in order to identify possible attacks. The "Cluster monitoring node election" approach as proposed in [6] relies on monitoring by an elected node, with election process occurring periodically. In accordance with the protocol description in [6], we simulated the election process as a recurring procedure where:

- Each node has its energy level decreased by a decrement corresponding to: (1) a random number generation - we assumed a hash-based random number generation; (2) the computation of a hash on the computed random number and the node identifier; (3) the sending of a message containing said identifier and hash; (4) the waiting for receiving all messages from neighbor (same-cluster) nodes; (5) the actual reception of all of these (n-1) messages (n being the

number of nodes within one cluster); (6) the verification of all received messages through the computation of (n-1) hashes; (7) the operation of the selection function; (8) the sending of a result message.

- As a result of the election, a new node becomes responsible of cluster monitoring. That is, the former monitoring node reverts back to the 'normal' behavior (sleeping mode, with periodic wake up) and the newly elected monitoring node switches to the 'monitoring' behavior (listen mode) for the next period.

4.3.2 Our Hybrid Solution

We implemented our proposed approach on the Csimu simulator by adding within the server processing operation a phase where the threat detection server periodically designates a new monitoring node within all clusters. To that aim, the server maintains a view of the network nodes statuses, which makes it able to designate for each cluster the node with the highest energy level as the new monitoring node. The process happens thus as follows:

- Upon waking up, each node sends to the threat detection server a message prompting for orders and informing the server about its current battery level. The server answers accordingly. The energy costs corresponding to message sending and response waiting, receiving and processing are taken into account and decremented from the nodes energy levels.

- Periodically, the server triggers a new designation of a cluster monitoring responsible node. This does not incur any immediate action. However, the subsequent request for orders of the former monitoring node and the newly designated monitoring node will make these learn, respectively, their orders to switch into 'normal' and 'monitoring' modes.

- The protocol that was implemented in the simulator introduced a slight variation as compared to the approach described above: each cluster is initialized with no active monitoring node. It is assumed that nodes in normal mode can pinpoint - even with much higher probabilities of false positives and false negatives - malicious behaviors and consequently warns the threat detection server, which will in turn mark the cluster as suspicious and assign a monitoring node within its population. The metrics for identifying clusters (proposed approach) / nodes (both approaches) as suspicious / compromised are explained in next section.

4.4 Simulation Parameters: Assumptions and Devices Characteristics

This section reviews the simulation parameters that are used in the simulation environment to rate nodes behaviors, on one hand, and to assess energy consumption, on the other hand.

4.4.1 Assessment of Attack Probability

Without implementing any actual attack detection scheme, we based our simulated threat detection function on a probabilistic model of attack detection, where each behavior (A_1, ..., A_n attacks, as well as the 'A_0' benevolence) can lead to a detection of any behavior within the (A_0,

A_1, ..., A_n) population, with different probabilities. This led us to define an attack recognition matrix M_{AR} as:

$$M_{AR} = \begin{pmatrix} p(D_{A_0}|A_0) & p(D_{A_1}|A_0) & \cdots & p(D_{A_n}|A_0) \\ p(D_{A_0}|A_1) & p(D_{A_1}|A_1) & \cdots & p(D_{A_n}|A_1) \\ \vdots & \vdots & \vdots & \vdots \\ p(D_{A_0}|A_n) & p(D_{A_1}|A_n) & \cdots & p(D_{A_n}|A_n) \end{pmatrix}$$

where $p(D_X|Y)$ corresponds to the probability to detect the event X knowing that event Y occurred.

In fact, two such matrices were defined corresponding to the two modes in which a node can be, respectively normal mode (non-monitoring node involved in usual node operations only) and monitoring node. The probabilities were tuned accordingly, to reflect the fact that a node in monitoring mode is much more accurate in its identifications, whereas the probabilities of false positives ($p(D_{Ai}|\ A_0) > 0$ for some i>0) and false negatives ($p(D_{A0}\ |\ A_i) > 0$ for some i>0)are much higher for a node in normal mode.

Based on this matrix, we were able to compute the probability of actual attack (whatever the A_j, j$\in$ [1; n]) occurrence upon the detection D_{Ai} of a given event A_i.

$$\begin{aligned} p(Attack|D_{A_i}) &= \frac{p(D_{A_i} \cap Attack)}{p(D_{A_i})} \\ &= \frac{p\left(D_{A_i} \cap (A_1 \cup \cdots \cup A_n)\right)}{p(D_{A_i})} \\ &= \frac{\sum_{k=1}^{n} \left(p\left(D_{A_i} A_k\right).p(A_k)\right)}{\sum_{k=0}^{n} \left(p\left(D_{A_i} A_k\right).p(A_k)\right)} \\ &= \frac{\sum_{k=1}^{n} \left(M_{AR}\left[k,\ i\right].p(A_k)\right)}{\sum_{k=0}^{n} \left(M_{AR}\left[k,\ i\right].p(A_k)\right)} \end{aligned}$$

In this computation, the elements M_{AR}[k, i] are the coefficients of the MR matrix defined above. The $p(A_0)$, $p(A_1)$, ..., $p(A_n)$ probabilities of occurrence of events A_0, A_1, ..., A_n are however other parameters that are required, and for which only approximate values can be used. In the simulator we used values derived from the probability of node compromise, and, for a compromised node, the probability of launching a type$i$ attack A_i. In actual conditions, where these probabilities of attacks cannot be precisely known, the system should use cognitive behavior where the optimal values of $p(A_k)$ would be statistically approached from the history of past detected events.

With an attack detection being mapped to the probability of actual attack occurrence, we were able to increase node compromise scores by a proportional coefficient. Eventually, we defined four thresholds t_1, t_2, t_3, and

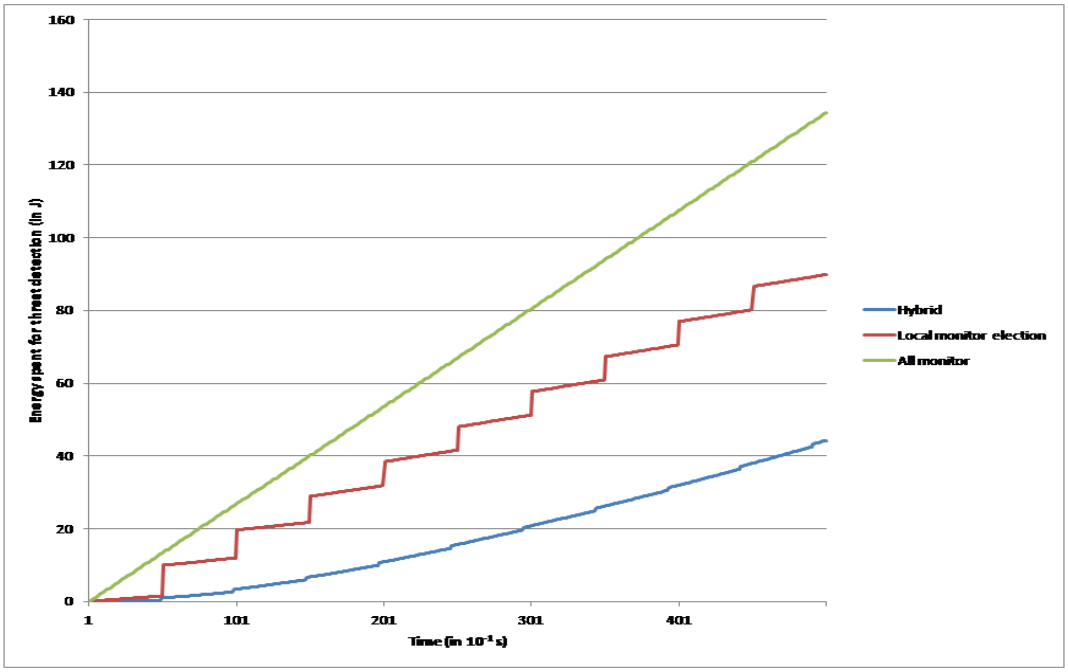

Figure 4: Energy consumption for threat detection operation in all three evaluated systems

t_4 respectively for marking a cluster as 'to be monitored' (when the cluster score exceeds t_1),marking a cluster as 'no longer to be monitored' (when the cluster score falls below t_2), marking a node as 'likely compromised' (when the node score exceeds t_3) and marking a node as 'likely benevolent' (when the node score falls below t_4[1]). We made the cluster and node scores decremented at each execution of the main loop, to mitigate false positives and let the system revert back to normal state after a compromised node has been excluded.

4.4.2 From Node Characteristics to Computation Parameters

We used the node characteristics listed in [11] corresponding to the following parameters: power in transmit mode (54 mW), power in receive mode (61 mW), power in listen mode (60 mW) and energy consumption for one processor cycle (8.64 nJ / cycle). Likewise, sensor nodes throughput was supposed to be equal to 75 kbps, as determined in [11].

Random number generation and hash function required in [6] were assumed to be based on the SHA-1 hash function, for which the required number of cycles per byte was computed, based on eBASH [2], to be 53.24 on the considered processor for a less than 64-byte long message (the shorter the message, the higher the cost for hashing one byte, due to the SHA-1 fixed operations that do not depend on message length).

[1]We did not use that fourth threshold, though in our simulation: instead, we considered that a compromised node was to be removed from the system; and could optionally be re-introduced after a while - but considered as an entirely new node.

4.5 Simulation Results

This section reviews the obtained results in terms of energy cost and efficiency, which prove that our proposed system, though slightly slower to react to malicious operations, is more energy-efficient than the other evaluated solutions.

4.5.1 Energy Consumption

Figure 4 presents the overall energy consumption during all processes related to threat detection for a 5-cluster, 50-node topology. This energy consumption is expressed in Joules per 10^-1 s for the three evaluated systems, namely 'All monitor' (all cluster nodes stay in monitoring mode), 'Elect' (there is only one monitoring node per cluster, which is regularly refreshed through an election carried out between the cluster members [6]) and our proposed 'Hybrid' scheme.

As was expected, the 'All monitor' approach is the most requiring in terms of energy with the 50 nodes (actually, 45 sensor nodes, since energy consumption is not measured on gateways by the simulation environment) consuming around 135 J in 50 s (a result that corresponds to the 60 mW that each node consumes in listening).

Figure 4 also highlights the high energy cost of [6], due to its heavy election processes. Even though only one node monitors a cluster at a time, the cost of election phases (the vertical transitions between plateaus in the curve) does not allow an overall energy gain higher than 33%, as compared with the 'All monitor' approach.

In order to highlight the difference between the local election phase of [6] and the server-assisted monitoring

node designation we proposed, we simulated in Figure 4 a synchronous approach of our hybrid approach, where all cluster nodes in all clusters perform mailbox checking (and, if required, mode change) at the same moment. Nevertheless, the vertical transitions are much smaller than those of [6], as could be expected from the lighter requirements exhibited by the mailbox checking, as compared to local election processing. Eventually, our solution performs approximately twice better than [6] and three times better than the 'All monitor' approach, with a final threat detection energy cost of only 44 J for 45 sensing nodes in 50 seconds (around 20W for each node). It has to be noted that the exponential-like allure of the 'Hybrid' system curve corresponds to the tuned behavior of the system, wherein clusters are initially not monitored (and consume almost no energy for threat detection) and start to get monitored only when compromised nodes reveal their malicious activities and are suspected by nodes. This leads to an acceleration of the energy consumption, which quickly reaches a cruising speed, where all clusters are monitored.

4.5.2 Threat Detection Reactivity

The behavior of all three simulated solutions with respect to compromised nodes identification is detailed below in Figure 5, Figure 6 and Figure 7.

Figure 5, Figure 6 and Figure 7 represent the compromise/react graph for each of the three simulated models. The dark, diamond-marked curves represent the number of unidentified compromised nodes within the system. The light, square-marked peaks represent detections of compromised nodes, and correspond therefore to a decrease of the other curve. Note that all three threat detection systems were tested in the same conditions, with compromising of the same nodes being triggered at the same time (scripted, non-random, compromising mode).

From these results, it appears that the 'All Monitor' approach performs the best with respect to speed in threat identification. Cluster election and the proposed hybrid approach, on the other hand, show almost similar results requiring more regularly a few seconds to detect and react to the presence of an attacking node.

4.5.3 Other Benefits of the Proposed Hybrid Approach

A synchronous embodiment of the proposed hybrid approach was depicted in Figure 4, in order to reflect the lower cost induced by the threat detection management phase (mailbox checking and required actions enforcement) as compared with that of the election approach. Beside this explanatory purpose, there is no need however to mandate synchronicity in our proposed solution: the sensor nodes may very well wake up at different times; the operation of the proposed approach would still work exactly the same. We simulated this asynchronous approach by explicitly requiring each sensor node to wake up at different times. We obtained equivalent performance with respect to reactivity to attacks; on the other hand, the resulting threat detection energy consumption curve presented, as expected, a smoother aspect (Figure 8).

Obviously, the support by our proposed approach of such asynchronicity (hence, of multiple sensor nodes that are not in phase regarding their sleep/wake up periods) could not be provided by the election approach, where all nodes within a common cluster must be involved in a synchronous way in the monitoring node election process. This support of sleeping node represents an important advantage of our proposed approach.

Finally, another benefit of our hybrid approach lies in the fact that interactions with a server entity can be exploited to further improve the threat detection system, by making it dynamically able to react to new threats. Indeed, the orders from the threat detection server to its clients could be extended to carry new attack signatures, thereby making the system both more evolutive and more in line with a given threat model. Simple extensions to the threat detection system, in line with the adaptive security approach, could lead to dynamically choosing which node(s) in a given cluster receive which attack signature(s).

5 Conclusion

This paper presents a threat detection system for industrial wireless sensor networks that could be qualified as hybrid in that it involves a semi-centralized process. The switch from normal threat detection mode to monitoring mode is triggered by the threat detection server, which bases on regular reports from nodes and updates its decision accordingly. We showed that this system, as reactive as the most studied state of the art solutions with respect to identifying threats, performs better from the viewpoint of energy consumption, saving around 50% of energy. The hybrid approach we propose, characterized by centralized management and local instantiation of threat detection, is also more flexible in terms of extension possibilities, opening up interesting development axes for the future of autonomous security in WSNs.

Acknowledgments

This work was financially supported by the EC under grant agreement FP7-ICT-258280 TWISNet project.

References

[1] F. Bao, R. Chen, M. J. Chang, and J. H. Cho, "Hierarchical trust management for wireless sensor networks and its applications to trust based routing and intrusion detection," *IEEE Transactions of Network Service*, vol. 9, pp. 169–183, 2012.

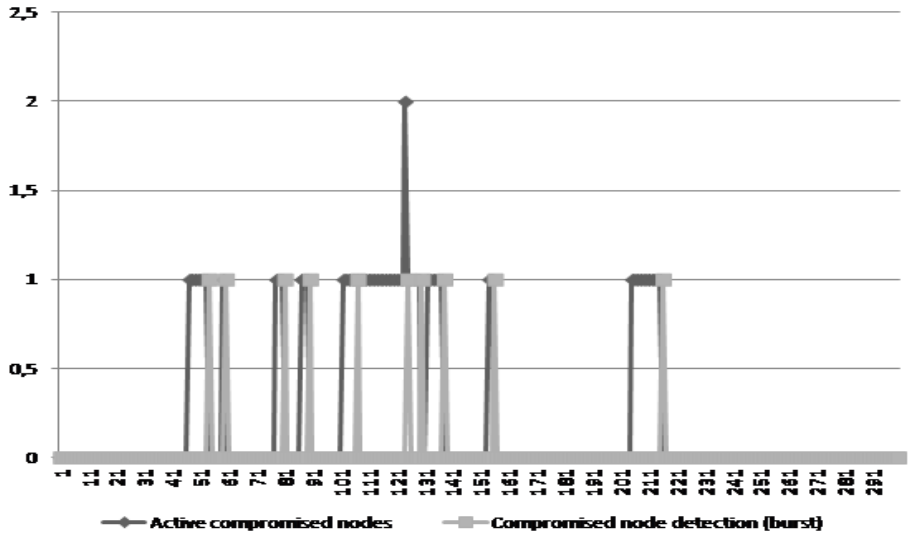

Figure 5: All monitor

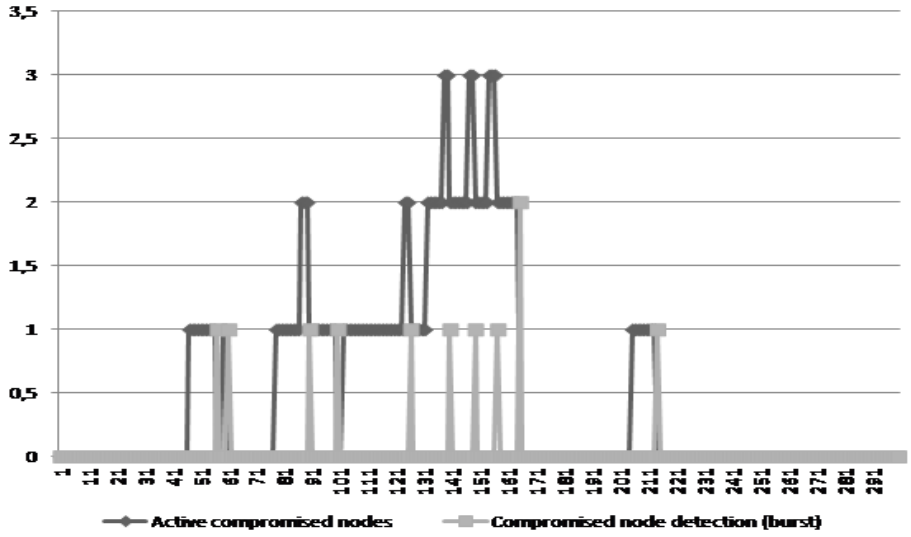

Figure 6: Local monitor election

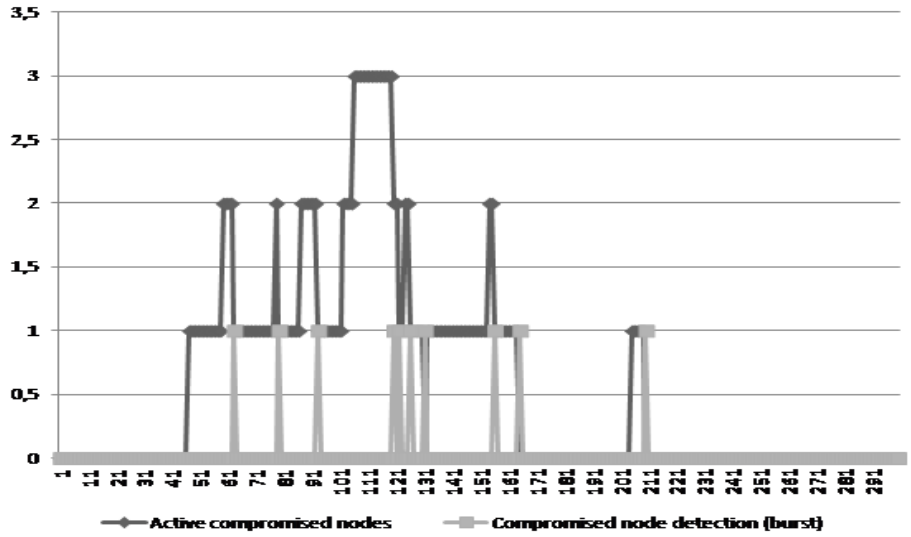

Figure 7: Hybrid approach

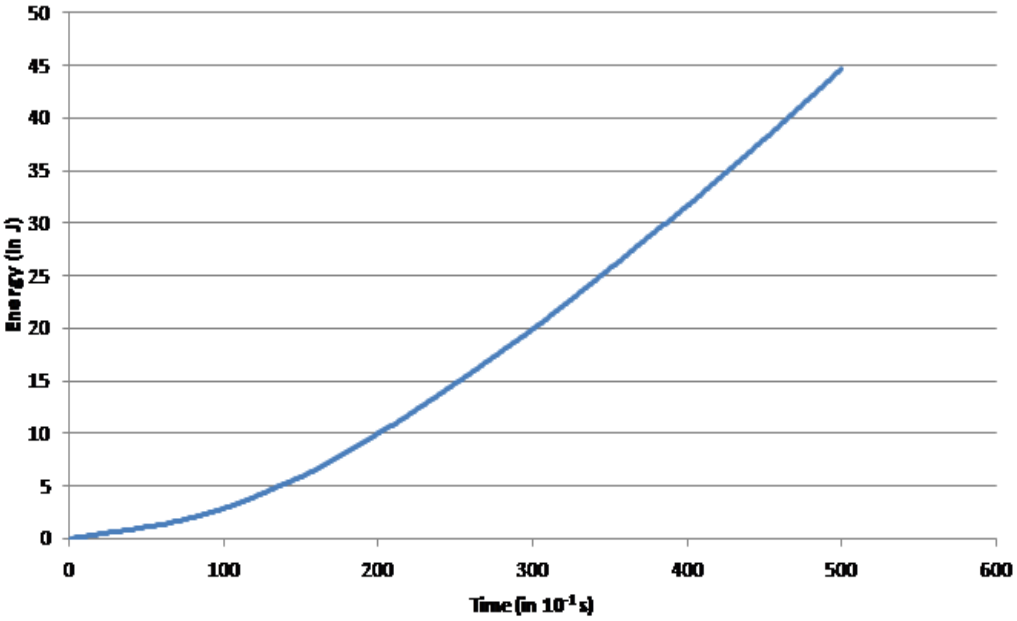

Figure 8: Energy consumption for threat detection operation in hybrid asynchronous mode

[2] D. J. Bernstein, *eBASH: ECRYPT Benchmarking of All Submitted Hashes*, 2013. (`http://www.bench.cr.yp.to/ebash.html`)

[3] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys and Tutorials*, vol. 16, pp. 266–282, 2014.

[4] A. P. da Silva, M. Martins, B. Rocha, A. Loureiro, L. Ruiz, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in *Proceedings of The First ACM International Workshop on Quality of Service and Security in Wireless and Mobile Networks (Q2SWinet'05)*, pp. 16–23, Montreal, Quebec, Canada, Oct. 2005.

[5] D. Elektronik, *Datasheet IEEE 802.15.4 node RCB230 V3.2*, 2015. (`http://www.dresden-elektronik.de`)

[6] Y. Huang and W. Lee, "A cooperative intrusion detection system for ad hoc networks," in *Proceedings of The First ACM Workshop on Security of Ad Hoc and Sensor Networks*, pp. 135–147, Fairfax, Virginia, Oct. 2003.

[7] K. Ioannis, T. Dimitriou, and F. C. Freiling, "Towards intrusion detection in wireless sensor networks," in *Proceedings of The 13th European Wireless Conference*, pp. 1–10, Paris, France, Apr. 2007.

[8] I. Krontiris, Z. Benenson, T. Giannetsos, F. Freiling, and T. Dimitriou, "Cooperative intrusion detection in wireless sensor networks," *Springer Wireless Sensor Networks Journal*, vol. 5432, pp. 263–278, 2009.

[9] I. Krontiris, T. Dimitriou, and F. C. Freiling, "Towards intrusion detection in wireless sensor networks," in *Proceedings of The 13th European Wireless Conference Computing*, pp. 1–7, Paris, France, Apr. 2007.

[10] R. Roman, J. Zhou, and J. Lopez, "Applying intrusion detection systems to wireless sensor networks," in *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC'06)*, pp. 640–644, Las Vegas, USA, Jan. 2006.

[11] Y. Ben Saied and A. Olivereau, "D-hip: A distributed key exchange scheme for hip-based internet of things," in *Proceedings of The First IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services (IoT SoS'12)*, pp. 1–7, San fransisco, USA, June 2012.

[12] R. A. Shaikh, "Group-based trust management scheme for clustered wireless sen-sor networks," *IEEE Transactions on Parallel Distributed Systems*, vol. 20, pp. 1698-1712, 2009.

[13] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public key cryptography for wireless sensor networks," in *Proceedings of The Third IEEE International Conference on Pervasive Computing and Communications*, pp. 324–328, Hawaii, USA, Mar. 2005.

[14] J. Zhang, "A trust management architecture for hierarchical wireless sensor networks," in *Proceedings of The IEEE Conference on Local Computer Networks*, pp. 264–267, Denver, Colorado, Oct. 2010.