MIT4WSN - Multipath Intrusion Tolerant Routing for Wireless Sensor Networks

João de Almeida, David Semedo, and Henrique Domingos
{jc.almeida, df.semedo}@campus.fct.unl.pt, hj@fct.unl.pt

NOVA-LINCS / DI/FCT/UNL
Laboratory for Computer Science and Informatics
Faculdade de Ciências e Tecnologia, 2829-516 Caparica, Portugal

Abstract. In this paper we present MIT4WSN a multipath redundant intrusion-tolerant routing system for wireless sensor networks (WSNs). MIT4WSN is composed of two combined mechanisms working in two different levels. In the first level the WSN is organized into a multi-tree structure, with disjoint redundant multipath routes discovered and established between sensor nodes and multiple base stations. In a second layer the multiple base stations implement an intrusion tolerant environment supported by byzantine fault-tolerant consensus mechanisms. We evaluate the proposal using a WSN simulation environment. In the evaluations we also use integrated tools, allowing the hybrid integration and calibration of the simulation environment including real sensor nodes and base-stations implemented with raspberry-pi single board computers.

Keywords: Wireless Sensor Networks (WSN), Secure Routing Protocols, Intrusion-Tolerance, Multipath Routing Protocols

1 Introduction

Sensor nodes are tiny, low-cost computing devices equipped with sensors of physical phenomena and wireless radio communication. Wireless Sensor Networks (WSNs) rapidly emerged as an important research area and a new approach to design scalable ad-hoc wirelessly internetworked pervasive computing systems. WSN nodes have important well-known limitations: (1) they can only communicate with their (short-range) neighbors by using specific radio-communication stacks (such as IEEE 802.15.4 [1] or 6LoWPAN (RFC 6282) over IEEE802.15.4 [2], requiring multi-hop routing services for scalability; (2) they have limited computational processing capacities and low memory resources; and (3) they have limited energy for autonomic operation, energy being a finite resource in some deployment scenarios and applications.

In many applications WSNs deal with critical data involving crucial monitoring and management of goods, lives, and livelihoods. In critical large-scale deployment scenarios, the nodes are exposed on the field, working autonomously, without any possibility of human supervision. In such cases, each sensor node is highly vulnerable to many kinds of attacks, both physical and logical, exploiting the intrinsic hardware/software limitations.

The design and implementation of realistic and effective security services for WSNs is not an easy task, considering the technological limitations [3].

These constraints limit the ability to perform computation-intensive tasks, such as public key cryptographic operations [4, 5], though Elliptic Curve Cryptography (ECC) offers a promising course of research. However, current ECC implementations still limit its applicability as a generalized solution [6]. Furthermore there is a lack of certified ECC implementations and ECC standards in the WSN industry solutions.

The relatively weak defenses of sensor nodes are susceptible to outsider attacks by much stronger adversaries equipped with more powerful computing and communication equipment (e.g., laptops equipped with 802.15.4 cards). But perhaps the most unique sensor nodes are usually ad-hoc distributed in the field in possible large-scale deployments with a considerable number of nodes in-situ. In these deployments, physical security conditions, supervision, or auditing facilities are not possible, contrary to what happens in most wired or ad-hoc wireless networks. As a result, unsupervised WSN nodes (including sensors, sync nodes, base-stations, or gateways) distributed in the coverage area are highly susceptible to possible physical compromises, being subject to intrusions, which endanger cryptographic keys and cryptographic-based communication protocols used to materialize authentication, confidentiality, integrity or access-control properties. Once compromised, the sensor nodes can then be easily exploited in order to introduce incorrect processing behaviors in the WSN, their goals being not only to cause incorrect sensing, but also to use the attacked nodes as vehicles of DoS, jamming, spoofing, or specific routing-service level attacks, at the several operation-levels of the typical 802.15.4 processing stacks.

Even considering the vast publication during the last few years, the design and implementation of realistic and effective security solutions for WSNs, namely intrusion tolerant solutions for secure routing services, is still an open topic.

In this paper we propose MIT4WSN - a multipath intrusion-tolerant routing system for wireless ad-hoc sensor networks (WSNs), using redundant routes as a strategy for dependability assumptions. Not following previous proposals that try to deal with the complexity of intrusion tolerance mechanisms and related coordination as processing components running in WSN nodes, the MIT4WSN design separates the intrusion tolerance solution in different mechanisms combined on two different approach layers:

- In a first layer the WSN (802.15.4 environment) is organized into a multi-tree structure, with disjoint redundant multipath routes discovered and established between constrained sensor nodes and multiple more resource-rich sync nodes or base stations (SN/BS)[1], operating as super-nodes operating in a overlaid layer of the WSN topology;
- On the second (overlaid) layer, the multiple SN/BS nodes are

[1] In the rest of the paper we use the acronym SN/BS to designate a node providing the combination of the conventional functionality as a sync node (SN) and base station (BS).

interconnected on a 802.11 WLAN, implementing an intrusion-tolerant environment supporting byzantine fault tolerance and running consensus protocols. This layer is used for routing coordination services, namely for cooperative agreements in the establishment of disjoint multipath routes, and also to agree on sensing values arriving through the multiple routes.

The main objective of the first layer is to prevent and tolerate damages caused by intrusions compromising the WSN nodes through a first line of defense built with lightweight mechanisms that can operate in a flexible way to address energy, reliability, and security tradeoffs. The second layer implements a complementary line of defense, providing protection from possible intrusion attacks that attempt to compromise the base stations.

Paper organization. The rest of the paper is organized as follows. Section 2 addresses the typology of WSN routing attacks, introducing the relevant approaches for intrusion-tolerance services related to the MIT4WSN solution as related work. Section 3 presents the network reference architecture, threat model, and the MIT4WSN design and its reference implementation stack. In section 4 we describe the components of the proposed solution. Section 5 is dedicated to the MIT4WSN implementation issues and experimental evaluation. Finally, section 6 concludes the paper.

2 Related work

WSN routing attacks. Several salient forms of threats on WSN routing protocols have been described in the literature. At the same time different types of security services and mechanisms have been proposed as possible countermeasures [7,8,9,10,11,12]. This paper is particularly focused on WSN routing attacks. For a taxonomy of classes of related attacks we consider a framework characterizing the routing attacks in the following typology, particularly addressed for pro-active routing protocols[2]: (i) attacks against the ad-hoc WSN discovery process and organization (ii) attacks against the selection and establishment of route paths, and (iii) attacks on the maintenance of previously established trust paths.

- **Attacks against the ad-hoc WSN organization and route discovery.** In this class we include: FRAs (Fake Routing Advertisements) and RREQ incorrect flooding [7,8] and Rushing attacks [9].
- **Attacks to the route establishment process.** In this category we include Hello Flood attacks [7], Synkholes [8], Wormholes [10] and Sybil Attacks [11,12].
- **Attacks against the maintenance of consistent routing paths.** This category includes attacks against the routing process itself, in the sequence

[2] In the context of the paper we consider pro-active (or table-driven) routing protocols, in which up-do-date consistent routing tables are inherent to previously established routes in each node, contrary to what happens in reactive (or on-demand) ad-hoc routing, where the nodes exchange routing information only when there are communications awaiting.

of the previous establishment of routes, such as: Blackholes [7], Byzantineholes and SPAM attacks. Blackholes are attacks against multi-hop routing where malicious nodes violate the assumption on the correct cooperation to forward received messages, as received, through an established route in which they are interior nodes, causing omission failures or message losses during the routing process. We define Byzantineholes as attacks induced from byzantine nodes that arbitrarily conjugate Blackholes (causing message omission faults) or arbitrarily modify payloads and incorrect routing decisions during the multi-hop routing process in the WSN mesh. SPAM are essentially DOS attack types caused by the generation of bogus messages sent by incorrect nodes to "kill" the sensors by exhausting the energy or to reduce the available network bandwidth.

Multipath routing strategy for secure routing. Data-link and MAC-level security services using lightweight cryptographic primitives to protect communications [13, 14, 15, 16] are not a final effective solution if we consider intrusions. Intruders are supposed to compromise nodes by capturing cryptographic keys, and they can still induce incorrect processing in the network. To deal with this, we must have complementary intrusion-tolerance mechanisms at the routing processing level as a component of network-level security services. Multi-hop routing is a fundamental service for the operation of WSNs, establishing end-to-end communication in the WSN mesh. Most of the security support for data aggregation, in-network processing, secure localization, intrusion detection, or key management relies on some secure routing scheme to exchange data and to maintain the correct network operation. Routing paths are usually established using a single path between source sensor nodes and SN/BS nodes. Although this scheme is well suited in WSNs where resources are limited, the failure of nodes or intrusions along the path would mean failure of the path and loss of data. Different approaches have been designed to address dependable routing schemes using a multipath strategy, in order to enhance the network availability, resilience and reliability, and aid in a timely critical decision-making by using the different available routes [17].

Intrusion tolerant multipath routing for WSNs. Despite that we find in the literature proposed related work approaches for secure multipath routing schemes for WSNs, in many approaches the previous identified issues are no well addressed. As stated in [17], a large number of secure multipath routing protocols doesn't address all the previous routing attacks and concerns and while the literature is also abundant in also discussing intrusion detection techniques for WSNs [19, 20, 21], the issue of how often these techniques can be efficiently used and implemented in real WSN nodes for pro-active intrusion tolerance guarantees (due to resource limitations and for energy reasons) is not explored in general. As another observation, there are some approaches for multipath intrusion tolerance using heterogeneous nodes as super-nodes, explored as resource rich cluster-heads and as more effective

solutions to achieve scalability, energy conservation, and reliability [18]. However the use of such solutions to address intrusion-tolerant sync nodes and base-stations, addressing byzantine fault tolerance services (as intended in this paper), is an unexplored topic as far as we know. It is relevant to notice that cluster nodes, sync nodes or base-stations are also susceptible of intrusion attacks in many deployment scenarios. However these nodes are always considered as trust-computing bases (out of scope of the adversary model definitions) in previously proposed solutions [18, 22, 23, 24]. Actually, the design of the major part of previously proposed multipath routing protocols [17,18] was previously addressed only as a reliability extension for tree-based routing protocols and the management of multipath routing for intrusion tolerance don't consider byzantine intrusion tolerant mechanisms and possible attacks to sync nodes or base-stations as addressed in the MIT4WSN design.

Byzantine intrusion tolerance. Although intrusion tolerance has been extensively studied in the past in the context of wired networks, it is still a target of recent relevant contributions. In [26,27,28] the authors address new relevant foundations for the support of byzantine fault-tolerance services for ad-hoc wireless networks, not only from theoretical assumptions but also dealing with real implementations and experimental assessment. We take these results as relevant seminal related work and inspiration sources for the MIT4WSN design and to implement the intrusion-tolerant services for the SN/BS overlay.

3 MIT4WSN design principles

3.1 Network architectural model

The reference network architecture that inspired the MIT4WSN design is represented in Fig. 1.

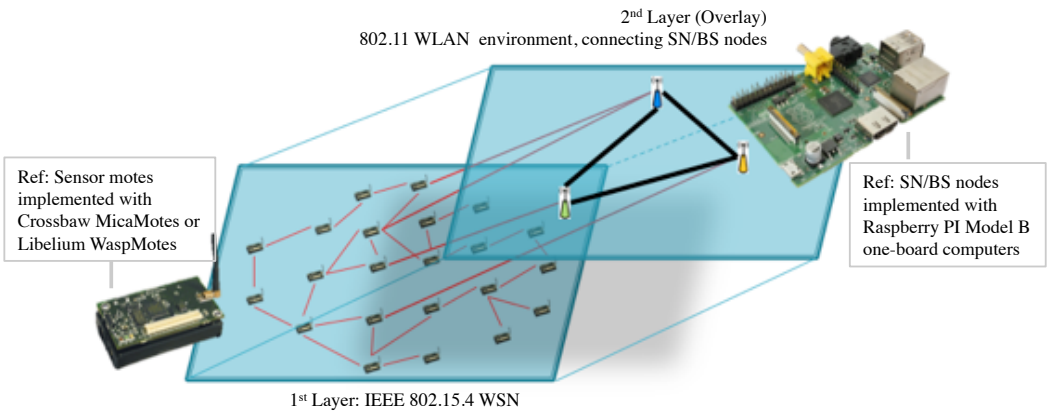

Fig. 1 MIT4WSN Network Reference Model

Sensor nodes organize themselves into a resilient tree-based multipath and multi-hop topology, collecting and forwarding sensor data (message payload

values) to root nodes (implementing SN/BS functionality). In the overall topology root nodes are seen as the cluster-heads of the established disjoint multi-hop trees. In our model the WSN is also regarded as a monitoring "island" and it can be interconnected to an Internet infrastructure (together with other "islands" by groups of BS/SNs).

As reference (also related to our implementation and developed prototypes) we show the typical nodes used to materialize the first and 2nd layers of the network architecture. BS/SN nodes are components implementing data aggregation activities, routing coordination and internetworking gateways, operating in typical structures or ad-hoc 802.11 WLANs. Inside the 802.15.4 WSN, the communication patterns are relatively simple compared to a traditional wired or an ad-hoc wireless network, as in general each WSN "island" is dedicated to one only application. Data transmission is dominated by local 1-hop communication between sensor neighbors, multi-hop forwarding between sensor nodes and BS/SNs and one-to-many from SN/BS nodes to sensors. Sensing data is sent from each sensor node to one or more base stations [7], through the disjoint paths available. We assume that the number of base stations per WSN is significantly less than the number of the WSN nodes (e.g., 4 to 13 BS/SNs for WSNs ranging from 100 to 1000 nodes as a scale reference). The base stations are relatively resource-rich in processing, storage, energy, and communication capabilities, implementing the TCP/IP stack, not suffering from the same constraints of used sensor nodes. The large number of resource-constrained sensor nodes and the small number of resource-rich base stations collectively form an asymmetric internetwork structure explored in the MIT4WSN design principles. The Fig 1 shows a network example using 22 sensor nodes (typically 802.15.4 crossbow micaZ motes[3] or Libelium WaspMotes[4]) and 3 SN/BS nodes (implemented with small credit-card sized one-board linux-enabled computers using the Raspberry PI model B technology equipped with 802.11 USB dongles and rutting the Raspbian Whezzy Linux distribution 3.1.0 [5]).

In the reference network model, we clearly identify two different layers: the first layer (as the 802.15.4 WSN itself), organized as a multi-tree structure using the disjoint redundant multipath routes discovered and established between sensor nodes and the multiple SN/BS nodes, and the second layer (802.11 WLAN environment) where SN/BS nodes operate as coordination supernodes for the first layer.

3.2 Adversary model assumptions

[3] MicaZ motes and datasheets are widely available from different vendors (e.g., http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf, ret. 19/Jul/2014)

[4] Information on Libelium WaspMotes are available from http://www.libelium.com (ret. 19/Jul/2014).

[5] Raspberry PI HW models and SW specifications are widely available in the Internet, e.g.: http://elinux.org/RPi_Hub, http://downloads.element14.com/raspberryPi1.html, http://www.raspbian.org, all links tested and retrieved on 12/6/2014.

In the MIT4WSN design and in its network reference model we assume the following adversary conditions.

In the first layer we consider possible outsider attacks trying to compromise the authentication, confidentiality, integrity and message freshness (by exploring message-replaying attacks), as well as intruders trying to compromise a certain number of WSN nodes to launch routing attacks, as previously defined in section 2. For the scope of this paper we only consider the attacks against the ad-hoc WSN organization and route discovery process. SPAM or DoS attacks or other attacks against the MAC-layer services are not considered in the scope of the paper.

In the second layer we assume intrusion attacks against BS/SN nodes. We consider that up to *f* nodes (in a set of a static group of *N* nodes) can exhibit a byzantine behavior (with *f* being a constant value during the system operation), failing arbitrarily or not following correctly the routing specification. Such byzantine nodes (processing as incorrect nodes) can become silent, inducing omission faults at this level, or they will try to send messages with arbitrary incorrect values. They can also collude to induce incorrect operation. The incorrect nodes don't follow the correct protocol specifications or algorithms that are subjacent to their coordination activities. The adversary model includes a dynamic omission failure-pattern in all message transmissions amongst correct BS/SN nodes, with safety requirements guaranteed even if the number of possible omissions is not necessarily limited during the communication rounds. However, we admit that for consensus termination purposes there is a limit in the number of omission faults. Then we consider that BS/SN nodes will eventually receive each correct message sent from a correct BS/SN node.

3.3 MIT4WSN design and implementation stack

The Fig. 2 represents the MIT4WSN software stack. The left side (MIT4WSN Layer 1) represents the MIT4WSN stack in WSN nodes (802.15.4 environment). The right side (MIT4WSN Layer 2) represents the MIT4WSN stack on BS/SN nodes (that are interconnected by IP (or IPSec) over 802.11 configurable for ad-hoc mode or infrastructure mode as possible configuration alternatives.

For the rest of the paper we consider the following as orthogonal services (out of the scope of the paper): KDS - a Key Distribution and Establishment Service to setup cryptographic keys and Security Associations for the MiniSec [13] implementation; IPSec KDS - a standard IPSec Key-Establishment Scheme (or ISAKMP Services reused from the IPSec Standard Stack); and the MiniSec protection layer, corresponding to our implementation of the MiniSec

stack [13] for TinyOS/Crrossbow TelosB and Libelium WaspMotes, ported from the initial implementation reference[6] and source code.

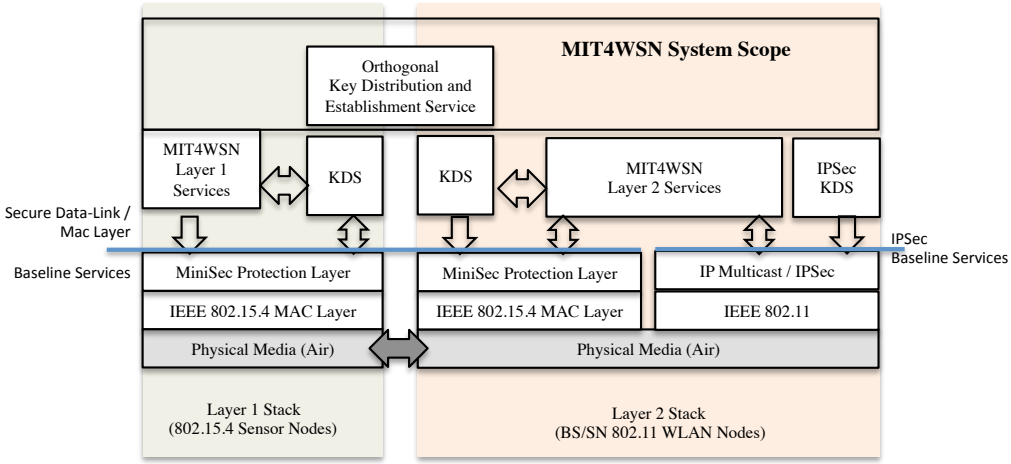

Fig.2 MIT4WSN Stack

4 MIT4WSN components and operation

We present now the MIT4WSN components. First we describe the base protocol operation at the WSN layer (or MIT4WSN Layer 1 Services) in Fig.2. Then we explain the multipath establishment process and the routing operation process. Later, we focus on the intrusion-tolerance services supported in the layer 2 (or MIT4WSN Layer 2 Services) in Fig.2.

4.1 MIT4WSN protocol in the layer 1

In order to explain the base protocol we will first introduce the network discovery and the selection of multiple disjoint routes. Initially the network has a random-based topology with nodes arbitrarily deployed in a certain geographic space. In this "scratch" deployment, some sensors will be naturally out of range of potential neighbors. During the network discovery phase, sensors try to find neighbors by sending HELLO-REQ messages, establishing pairwise neighboring associations, with symmetric communication patterns. Some sensors will discover BS/SNs as neighbor nodes, and all the pairwise associations will form a flat network coverage pattern (as a coverage graph). Remembering, all the messages exchanged by sensors or BS/SN nodes are protected by a subjacent secure data-link layer protocol based on the Minisec stack [13] enhanced with a dynamic pairwise key-establishment scheme for sensor neighboring associations. All the messages are primarily protected by an OCB encryption scheme using AES (with pre-established 128 bit keys, and AES-based CMACs with previously established MAC keys) for message authentication guarantees. All sensors have the Minisec cryptographic keys

[6] See https://sparrow.ece.cmu.edu/group/minisec.html for more details.

pre-established by the KDS service, with the keys shared between sensors and each one of the BS/SN nodes. Summarizing, each sensor has an initial secret key shared with the BS (and we suppose that each BS previously knows all the identifiers of all the sensors in the network). For message authenticity confidentiality, integrity and freshness control, each sensor initially derive a MAC key from pre-established secrets (to compute and validate the CMAC codes) and a pairwise confidentiality key (for AES 128), using a generation process in which those keys are generated from the initial shared secret seeds by means of secure hash computations.

For the purpose of the following explanation and due to the paper size limitation, we consider that the key-establishment process for the Minisec setup is secure, with the same considerations as discussed in [13]. Then the route discovery process builds the desired topology of the network setting up routing tables at each node, over this protection layer. The process requires five steps as follows.

Step 1. Each BS/SN (bs_{ID}) floods a REQUEST-REQ (or RREQ) message trying to reach all the sensors connected with some neighbor in the network coverage. The message has a tuple $\langle RREQ, bs_{ID} \rangle$ and it is propagated epidemically.

Step 2. Each sensor node will send a ROUTE FEEDBACK RESPONSE message (FDBK) to its observed neighboring topology. When a node n_i receives a RREQ message, it checks if it has already received the message. If the message is new, the emitter is added to the neighbors' set (N_i) and defined as father of i; if not, the emitter is simply added to N_i after a certain amount of time, and n_i sends to each bs_{ID} its neighborhood data; These data are sent through the inverse paths (with each node sending to its father, until reach the bs_{ID}). This message has the $\langle FDBK, i, N_i \rangle$ format.

Step 3. Each base station receives the ROUTE RESPONSE messages sent by each sensor back to each BS, and it computes all the observed routes. For a typical network of up to 1000 nodes, each BS/SN can easily compute and maintain routing entries, indexing these entries by each sensor's unique identifier. For the fourth step, each BS/SN will compute the topology information to obtain the candidates to multipath forwarding tables in its local observation, and securely multicasts to the other BS/SNs the multipath routes locally computed for each node. In this process, each BS/SN merges the multipath routes to build a global vision of all the paths established for each BS. For this purpose they use a fault-tolerant consensus protocol to agree on the global vision of the multipath selection (using the mechanisms explained later, in the section 4.4).

Step 4. In this step each BS/SN unicasts the previous agreed tables to the respective nodes. During the process, the network topology evolves to form multipath trees, where the complete multipath trees are the union of all the disjoint multiple trees from each sensor to each BS/SN. The messages with

the routing tables (called *RUPD* messages) are sent to the nodes by each BS/SN and these messages are ordered by distance between the given node and the BS/SN, to allow a gradual optimized dissemination of all the announcements. This way, routing tables from nearer nodes can be used to route the routing tables of the farther ones.

Step 5. After the reception of their routing tables, the WSN nodes install the local routing tables, with a configurable threshold timeout. After the timeout the network is finally self-organized and ready to operate.

4.2 Intrusion tolerance services in the layer 2

This phase starts when a message reaches a BS/SN, and it depends on the routing parameterization mode. When a message is routed through only one route to one BS/SN, the BS/SN validates the message and if it is valid the message is delivered to the application level. This validation verifies the message authentication, confidentiality, integrity and freshness (according to the MiniSec security validations). When a message reaches a given BS/SN through the several routes, the BS/SN stores all the different replicas of the message. In the current implementation, when the consensus is reached and when the BS/SN has stored more than half of the replicas with the same value, the message is considered as validated (with the agreed value). At this point, there are two possibilities: the message is just destined to that one BS, and after the validation, the message is delivered to the application level; or the message has been sent to several BS/SNs and it is necessary to start a consensus involving all the BS/SNs for the agreement on the validated data. It is important to understand that if the BS/SNs agree on a given sensing value (in a received message), it is considered as a correct value and it will be delivered as "agreed" to the application level by all the correct BS/SNs. If a consensus is not reached, the message is simply ignored.

Byzantine fault tolerance and data consensus support for messages exchanged by the layer 2 nodes are implemented using a support stack with two protocols: MVC/BC (or multi-value message consensus built on top of an intrusion tolerant binary consensus). These part of the MIT4WSN implementation is strongly based on the previous relevant work on the implementation of the Turquois protocol [26] and inspired by the lessons and remarks in [27, 28].

Binary Consensus. BC follows a variant of the Turquois protocol specification [26] adapted to run on top of a JVM and Java framework and over IPSec on raspberry Pi nodes. Turquois was originally designed as a randomized binary consensus protocol allowing k processes out of n ($k \subseteq n$) reach a binary consensus $v \in \{0,1\}$. The correctness of the protocol is guaranteed as long as the Byzantine flaws f satisfy the condition $f < n/3$. Details for the BC implementation is available in [26].

Multi-Value Consensus Layer. MVC is an implementation of a multi-value consensus protocol that reuses the BC. MVC is implemented in the

MIT4WSN stack as an asynchronous and probabilistic fault-tolerant protocol, designed to reach consensus over a set (configurable from 1 to max. 10) messages exchanged by the SN/NS nodes, maintaining the following properties even under dynamic omission and byzantine failures.

- *Validity*. If all SN/BS nodes propose a message and payload for consensus, then any correct node that decides, decides with the same payload.
- *Agreement*. Two correct SN/BS nodes will never decide differently about the payloads they consider correct.
- *Termination*. At least k SN/BS nodes performing correctly, eventually will decide, with probability 1.

The messages are initially pre-processed by each BS/SN in the following format: <sender nodeID, SHA1(message-payload)>. Our implementation is designed to reach consensus over the set of these messages. The protocol offers intrusion and fault tolerance in the presence of f failures or attacks whenever we can satisfy the condition $f < n/3$. Considering the asynchronous nature, some communications can fail without a compromise in the system operation. The MVC protocol allows the support for the agreement primitive for data consensus over any type of values since messages are computed as byte arrays. The MVC/BC stack is supported by configuration on a set of different base communication back-ends supported on BS/SN nodes: IP Multicast over 80211 in AdHoc or Infrastructure modes, IP Multicast over IPSec in transport mode, and a FIFO ordered multicast protocol implemented on top of TCP persistent connections supported on TLS/TCP.

5. Implementation and evaluation

We implemented the MIT4WSN stack, as previously described for the network reference model in the section 3. From different extensive evaluations of the MIT4WSN protocol, and given the paper size limitations, we present some assessment results about the MIT4WSN resilience under intrusion conditions. In our evaluations we used a WSN simulation environment with integrated tools for the hybrid integration and calibration of real sensor nodes implemented with Crossbaw MicaZ or WaspMote sensors and BS/SN nodes implemented with raspberry-pi single board computers. Real nodes can be integrated as “virtual nodes” in the simulation environment, using the facilities provided by SecWSNsim. In the simulator, the messages sent to those virtual nodes are really received by the external real nodes that execute the correspondent code in the simulated nodes. Messages sent by the real nodes are sent to the simulation environment, and forwarded to the simulated neighbors, as coming form any other simulated node. The MIT4WSN implementation is highly configurable, namely regarding the policy to disseminate messages in WSN nodes, parameterization of message transmission scheduling policy, the number of sensor nodes or BS/SN nodes, as well as the base communication backend. For the following results we used

the SecWSNsim/WiseNet simulation environment[7] to simulate the MIT4WSN layer 1 stack and real Raspberry Pi nodes (Model B) running the Raspbian OS (Linux Distribution) implementing BS/SNs for the layer 2 (integrated as an hybrid real and simulation environment). We used parameters obtained from running MIT4WSN on six Crossbow MicaZ Motes to calibrate the simulation environment for large-scale simulated networks. To compare our results we implemented from the MIT4WSN design components an extended version of the INSENS protocol (called MINSENS++) comparing MINSENS++ with the INSENS implementation [22].

Resilience under attack conditions

In these experiments we generated WSN random topologies in the simulation platform for 300, 500 and 1000 nodes, represented in Table 3. As shown in the table 3, for each generated WSNs we selected 10% of nodes as sender nodes, a number of BS/SNs for MINSENS ++ for each WSN size (remembering that we use only one BS/SN for INSENS). The table 3 also shows the average number of established disjoint routes observed for the different settings of MINSENS++.

WSN Size	# of senders	# of Base Stations	# of disjoint routes for each sender to different BS (average)	# of disjoint routes between each sender and each BS (average)
300	30	4	9.2	2.25
500	50	7	23.1	3.29
1000	100	10	35.9	3.60

Table 3. Parameterizations

From the parameterizations in table 3 we configured a certain number of nodes (varying arbitrarily in each experiment from 10% to 20% of the total WSN nodes, not considering sender nodes) to simulate omission or byzantine behavior. For each setting we observed 10 executions of MINSENS++ and INSENS for each intrusion behavior. For the observations, we also include in the MINSENS++ settings a number of intruders in the total of BS/SN nodes, in the following way: 1 intruder for 4 BS/SN nodes, 2 intruders for 7 BS/SN nodes, 3 intruders for 10 BS/SN nodes and 4 intruders for 13 BS/SN nodes. The reliability is evaluated as the number of messages after the BS/SN consensus over the total of messages sent by correct WSN sender nodes.

[7] The simulation environment and simulation tools are available in http://asc.di.fct.unl.pt/SITAN/prototypes/prototypes.html.

WSN Size	# of BS/SN Nodes	WSN connectivity (covered nodes)	Reliability with Omission Faults	Reliability With Byzantine Faults
300	4	100%	98% / 64%	99% / 62%
500	7	97%	95% / 72%	93% / 61 %
1000	10	91%	89% / 73%	77% / 59 %
1000	13	95%	93% / 66 %	89% / 57 %

Table 4. Connectivity and reliability metrics in the randomly generated topologies, showing the intrusion-tolerance effectiveness of MINSENS++ compared to INSENS

The obtained average metrics are shown in the table 4. The reliability comparison MINSENS++ / INSENS is represented in columns 3 and 4. As shown, MINSENS++ exhibit a significantly higher resilience against the simulated intrusions, compared with INSENS running only with one correct BS/SN node.

For the results shown in table 4, in our experiments each sender sends 1000 messages over IEEE 802.15.4, each message having a size of 32 Bytes. The messages have a header (28 bytes) and payload (4 Bytes). Although the IEEE 802.15.4 standard states that a message can have up to 127 bytes, we observed that messages with more than 56 bytes would cause many collisions with real sensors. In our observations we use a payload of 4 bytes, representing a (32 bits) integer corresponding to a sensing value. Measuring the real throughput of IEEE 802.15.4 communications between a pair of real sensor motes (with ping tests of 802.15.4 packets), a value of 2.2 Kbps was achieved as average, so in SecWSNsim/WiSeNet simulator was thus tuned to 1.1 Kbps (message scheduling rate) in order to obtain more approximated results from the reality. After such calibration, the *end-to-end* average latencies for different network sizes (from 300 to 1000 sensor nodes) were measured by an external coordination process reading values from the BS/SN nodes; the obtained results are represented on table 5.

WSN Size	# of hops per sender (average)	*k-resiliency* factor	MINSENS++ routing latency (milliseconds)	MVC consensus latency, Intrusion Free (milliseconds)	MVC consensus latency, Intrusion Stop (milliseconds)	MVC consensus latency, Byzantine Attack (milliseconds)	% of messages with terminated consensus
300	7.7	3	1405	354	1298	2098	96%
500	10.3	5	1790	666	2586	3134	88%
1000	11.9	7	1968	1441	3526	3852	75%

Table 5. Solution effectiveness

The table 6 shows the effectiveness evaluation of the integrated MIT4WSN solution, also showing the overall impact of the components in the two layers of the solution. This experiment allowed us to verify that the integration of the two main components of the intrusion tolerant routing service (routing layer at the WSN level) and intrusion tolerant consensus layer (provided at the SN/BS level) is a good solution.

WSN Size	Attacker Type	End-to-end latency (millisen-conds)	Effective throughput	Time consumed on routing layer (%)	Time consumed on consensus layer (%)
300	Intrusion Free	1759	1.45 Kbps	80%	20%
	Intrusion Stop	2703	0.95 Kbps	52%	48%
	Byzantine Attack	3503	0.73 Kbps	40%	60%
500	Intrusion Free	2456	104 bps	73%	27%
	Intrusion Stop	4376	58.5 bps	41%	59%
	Byzantine Attack	4924	52 bps	36%	64%
1000	Intrusion Free	3409	75 bps	58%	42%
	Intrusion Stop	5494	46.6 bps	36%	64%
	Byzantine Attack	5820	44 bps	34%	66%

Table 6. Effective throughput and bottlenecks of components

The system works as expected, routing the data generated by the WSN sensors to the multiple BS/SN nodes, and the assessment results show that the solution is resilient against different types of attacks (Failure Free, Intrusion Stop and Byzantine Attacks) on WSN nodes and BS/SN nodes. According to the observed throughputs, the bottlenecks depend on the number of SN/BS nodes performing the consensus protocols. For 7 Base Stations the bottleneck changes from the routing overhead (on the layer 1) to the consensus components (on the layer 2).

6. Conclusions

The paper presents MIT4WSN, a tree-based multipath redundant intrusion-tolerant routing system for wireless sensor networks (WSNs). MIT4WSN combines two different intrusion-tolerant mechanisms running at two different approach levels: WSN level (or IEEE 802.15.4 environment) and BS/SN level, as an overlaid 802.11 WLAN environment. At the WSN level, the sensor nodes are organized in a multi-tree structure, with disjoint multipath routes established between the WSN nodes and multiple BS/SN nodes. At the second level these multiple BS/SN nodes implement a byzantine fault-tolerant environment supporting intrusion tolerance with consensus protocols used for routing coordination and management purposes, and to protect WSN data-dissemination from possible attacks against the sensor nodes and against the BS/SN nodes themselves. With the proposed solution, we designed an intrusion-tolerant routing service that avoids the possible damage caused by intruders compromising the deployed sensor nodes or BS/SN nodes, to inject, discard,

modify, or block correct data packets sent by correct WSN nodes, and, at the same time, we also avoid possible damages caused by adversaries that try to compromise the base stations and sync nodes.

The MIT4WSN implementation and evaluation based on simulations, implementation prototypes, and test bench installations, shows that the proposed solution is promising and valid. The achieved results consolidate the potential of innovation that may be explored from the proposed idea and open interesting future research work directions for more extensive evaluations form the current design and implementation, as well as, possible optimization options.

References

1. http://www.ieee802.org/15, IEEE 802.15.4 Working Group for WPAN, ret. 17/June/2014
2. *IPv6 over Low Power WLAN* (6lowpan), IETF 6lowpan Group http://datatracker.ietf.org/wg/6lowpan, ret. 17/Jun/2014
3. João Almeida, *Intrusion-Tolerant Routing for WSNs Using Data Consensus Primitives*, MSc Thesis, DI-FCT-UNL (July 2013)
4. A. Perrig, R. Szewczyk, V. Wen, D. Culler, J. Tygar, *Spins: Security protocols for sensor networks*, Wireless Networks Journal WINET 8(5) (2002)
5. J. Deng, R. Han, S. Mishra, *The performance evaluation of intrusion-tolerant routing in wireless sensor networks*, In IEEE 2nd International Workshop on Information Processing in Sensor Networks (IPSN‟03), Palo Alto, CA, USA (April 2003)
6. P. Szczechowiak, L. Oliveira, M. Scott, M. Collier, R. Dahab, *NanoECC: Testing the Limoits of Eliptic Curve Cryptography in Sensor Networks*, 5th European Conference on Wireless Sensor Networks, LNCS 4913 pp. 305-319, Springer-Verlag (2008)
7. Y. Wang, Y. Tseng, *Attacks and Defenses of Routing Mechanisms in AdHoc and Sensor Networks*, Security in Sensor Networks, pp 3-25, Auerbach Pub., Ed. Y. Xiao (2007)
8. A. D. Wood, L. Fang, J. A. Stankovic, and T. He, *Sigf: a family of configurable, secure routing protocols for wireless sensor networks*, in *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, SASN ’06, pp. 35–48 (2006)
9. Y. Hu, Adrian Perrig, D. Johnson, *Rushing attacks and defense in wireless ad hoc network routing protocols*, 2nd ACM Workshop on Wireless Security WISE2003, San Diego, CA, USA (September 2003)
10. Y. Hu, A. Perrig, D.B. Johnson, *Packet leashes: A defense against wormhole attacks in wireless networks*, In Proceedings of IEEE INFOCOM 2003 (April 2003)
11. J. Douceur, *The sybil attack*, 1st International Workshop on Peer-to-Peer Systems, LNCS volume 2429, Cambridge, MA, USA (2002)
12. J. Newsome, R. Shi, D. Song, A. Perrig, *The sybil attack in sensor*

networks: analysis and defenses, In Proceedings of IEEE International Conference on Information Processing in Sensor Networks - IPSN 2004 (April 2004)
13. M. Luk, G. Mezzour, A. Perrig, V. Gligor, *MiniSec: A Secure Sensor Network Communication Architecture*, ACM IPSN 2007, USA (2007)
14. Chris Karlof, Naveen Sastry, and David Wagner. *TinySec: A link layer security architecture for wireless sensor networks*. Second ACM Conference on Embedded Networked Sensor Systems - SenSys (2004)
15. *Zigbee specification*. Technical Report Version 1.0, ZigBee Alliance (June 2005)
16. Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. *SPINS: Security protocols for sensor networks*, 7th Annual International Conference on Mobile Computing and Networks, MOBICOM (2001)
17. E. Stravou, A. Pitsillides, *A Survey on Secure Multipath Routing Protocols in WSNs*, Elsevier Computer Networks, Vol. 54, Nr 13 (2010)
18. H. Al-Hamadi, I. Cheng, *Redundancy Management of Multipath Routing for Intrusion Tolerance in Heterogeneous Wireless Sensor Networks*, IEEE Trans. on Network and Service Management, Vol. 10, Nr 2 (2013)
19. S. Bo, L. Osborne, X. Yang, and S. Guizani, *Intrusion detection techniques in mobile ad hoc and wireless sensor networks*, IEEE Wireless Communication Mag., vol. 14, no. 5, pp. 560–563 (2007)
20. I. Krontiris, T. Dimitriou, and F. C. Freiling, *Towards intrusion detection in wireless sensor networks*, European Wireless Conference (2007)
21. J. H. Cho, I. R. Chen, and P. G. Feng, *Effect of intrusion detection on reliability of mission-oriented mobile group systems in mobile ad hoc networks*, IEEE Trans. on Reliability, vol. 59, no. 1, pp. 231–241 (2010)
22. J. Deng, R. Han, S. Mishra, *Intrusion Tolerant Routing for Wireless Sensor Networks*, Computer Communications, Vol. 29, Nr 2 (2006)
23. J. Deng, R. Han and S. Mishra, *Intrusion Tolerant Routing for Wireless Sensor Networks*. Elsevier Journal on Computer Communications, Special Issue on Dependable Wireless Sensor Networks, 29(2) (2005)
24. W. Lou and Y. Kwon, *H-spread: A hybrid multipath scheme for secure and reliable data collection in wireless sensor networks*, IEEE Transactions on Vehicular Technology, vol. 55, pp. 1320–1330 (2006)
25. SS.-B. Lee and Y. Choi, *A secure alternate path routing in sensor networks*, Computer Communications, vol. 30, pp. 153–165 (2006)
26. H. Moniz, N. Neves, M. Correia, *Turquois: Byzantine Consensus in Wireless Ad hoc Networks*, DSN 2010: 537-546, Chicago, USA (2010)
27. H. Moniz, N. Neves, M. Correia, *Byzantine Fault-Tolerant Consensus in Wireless AdHoc Networks*, IEEE Trans. Mobile Computing, 12(12):2441-2454 (2013)
28. H. Moniz, N. Neves, M. Correia, Paulo Verissimo, *RITAS: Services for Randomized Intrusion Tolerance*, IEEE Trans. on Dependable Sec. Computing, 8(1):122-136 (2011)