Attack Detection Using Evolutionary Computation

Martin Stehlik, Vashek Matyas and Andriy Stetsko

Abstract Wireless sensor networks (WSNs) are often deployed in open and potentially hostile environments. An attacker can easily capture the sensor nodes or replace them with malicious devices that actively manipulate the communication. Several intrusion detection systems (IDSs) have been proposed to detect different kinds of active attacks by sensor nodes themselves. However, the optimization of the IDSs w.r.t. the accuracy and also sensor nodes' resource consumption is often left unresolved. We use multi-objective evolutionary algorithms to optimize the IDS with respect to three objectives for each specific WSN application and environment. The optimization on two detection techniques aimed at a selective forwarding attack and a delay attack is evaluated. Moreover, we discuss various attacker strategies ranging from an attacker behavior to a deployment of the malicious sensor nodes in the WSN. The robustness of the IDS settings optimized for six different attacker strategies is evaluated.

Keywords Attacker strategy · Evolutionary algorithm · Intrusion detection system · Multiobjective optimization · Wireless sensor network

1 Introduction

Wireless Sensor Networks (WSNs) are highly distributed networks often deployed in open or even hostile environments. Sensor nodes are usually small low-cost and resource constrained devices with no tamper resistance employed and can be easily captured by an attacker. Furthermore, malicious devices considered as benign by other sensor nodes can be deployed in the network. Possible attacks on WSNs range from passive eavesdropping where an attacker listens on the ongoing traffic in promiscuous mode to active interfering and manipulating of the communication.

M. Stehlik (✉) · V. Matyas · A. Stetsko
Masaryk University, Brno, Czech Republic
e-mail: xstehl2@fi.muni.cz

In this chapter, we aim at the detection of such active attacks by sensor nodes themselves with respect to restricted capabilities of the sensor nodes. Each sensor node can be equipped with an intrusion detection system (IDS) [1]. Thus, an entire network area can be monitored for malicious behavior in a distributed manner. Several detection techniques have been proposed to detect various kinds of attacks on WSNs. Unfortunately, many of them are proposed for a specific case or their optimization is left unresolved. Moreover, incorporating an IDS brings necessary additional demands on sensor nodes' resources. In our work, we aim at automatic configuration of these detection techniques for a specific application scenario, topology and environment, using evolutionary computation having both the IDS accuracy and resource consumption in mind.

In this work, we demonstrate the functionality of our optimization framework consisting of a simulator and an optimization engine utilizing multi-objective evolutionary algorithms on two detection techniques proposed in [2]. The detection techniques are aimed at two kinds of active attacks—the selective forwarding attack and the delay attack. Our optimization framework provides Pareto front approximations consisting of different IDS settings with respect to three objectives—false positives, false negatives and memory consumption. We elaborate on these IDS settings found by evolution. Furthermore, we discuss various attacker strategies that can be used by an attacker and discuss the robustness of the IDS settings found for a specific attacker strategy in cases where another attacker strategy is used.

The contributions of this work are the following:

1. We provide a complex optimization framework for optimization of IDSs for WSNs. Various multi-objective evolutionary algorithms (NSGA-II, SPEA2, IBEA) can be used for optimization. The optimization framework can be easily extended to solve another optimization issue in WSNs.
2. We demonstrate the functionality of the optimization framework through extensive experiments on detection techniques detecting two different attacks—selective forwarding and delay attack.
3. We discuss and evaluate various attacker strategies. The IDS is optimized for six different attacker scenarios. Robustness of optimized IDS settings for each of the attacker strategy is evaluated on every other attacker strategy.

The chapter is organized as follows. In Sect. 3, we present our intrusion detection system and describe detection techniques used to detect the selective forwarding attack and the delay attack. Our optimization framework utilizing evolutionary algorithms and the metrics we use to compare resulting populations are described in Sect. 4. Various attacker strategies that can be used by an attacker are discussed in Sect. 5. In Sect. 6, we specify the settings of the evaluated IDS and of the experiments. Also, our test case WSN is presented. Experiment results are elaborated upon in Sect. 7. Related work is discussed in Sect. 2 and the chapter is concluded in Sect. 8.

2 Related Work

The related work is divided into two parts. First, we discuss related detection techniques for WSNs. Second, we discuss computational intelligence-based solutions for IDSs in WSNs.

2.1 Selective Forwarding and Delay Attacks

Selective forwarding attack has been among the most discussed attacks in WSNs during recent period. Karlof and Wagner [3] introduced selective forwarding attack in WSNs and discussed the possibilities of an attacker to place a malicious sensor node on a path between data source and base station. da Silva et al. [1] defined a "retransmission rule" as listening to a packet by an IDS whether it was forwarded by monitored sensor node or not. Krontiris et al. [4] set a threshold value for the percentage of packets dropped to 20 %. Another proposals of detection techniques for selective forwarding attack where the parameter setting is left unresolved can be found, e.g., in [5, 6]. To the best of our knowledge, no work has been published on such a complex parameter optimization for collaborative detection of selective forwarding attack.

The delay attack detection has not been discussed very much. da Silva et al. [1] defined a "delay" rule as a timeout before which a retransmission by monitored sensor node has to occur. Liu et al. [7] used the forwarding delay time measurement for their complex insider detection technique but its parametrization was left unresolved. To the best of our knowledge, we are the first who present a complex collaborative detection technique aimed at the delay attack detection.

2.2 Computational Intelligence-Based Solutions

A few papers utilize computational intelligence-based solutions to secure WSNs with IDSs.

Khanna et al. used single-objective evolutionary algorithms for several optimization issues in WSNs [8–10]. In [8], the authors aimed at minimizing power consumption while maximizing the coverage and exposure by switching the sensor nodes to the following states: (1) *inactive sensor node*; (2) *active sensor node*; (3) *cluster-head*; and (4) *inter-cluster router*. In [9], Khanna et al. presented an approach on a WSN deployment how it can be optimized dynamically and considered from the security point of view as well. The authors considered deployment of cluster-heads and inter-cluster routers allowing encryption and authentication, respectively. Finally, in [10], Khanna et al. incorporated *local monitoring nodes* into the network that observed suspicious behavior like data message patterns, message collisions and sensor positioning in their neighborhood. An IDS placement problem was addressed by adding evaluation of *local monitoring nodes* to the fitness function used in [9]. In all papers [8–10], all the objectives were blended into a single fitness function.

To the best of our knowledge, Heady et al. were the first introducing evolutionary algorithms to the area of IDSs for wired networks and the work of Khanna et al. [10] is the only work on optimization of IDSs [11] using evolutionary algorithms for WSNs except of our IDS optimization framework. We introduced multi-objective evolutionary algorithms to IDS in WSNs in [12].

Several works on another metaheuristics utilized for IDSs in WSNs can be found. Banerjee et al. [13, 14] used swarm intelligence—ant colony—for localization of the source of intrusion. The ants traverse the sensor nodes via edges that connect the neighboring sensor nodes. Adjacent sensor nodes with maximum number of violations represented as pheromone are preferred. When an ant visits an edge, the application of the local update rule makes the edge pheromone level diminish to the edge becoming less attractive. Mukherjee and Sen [15] detect intentionally sent erroneous data on the base station using neural networks. A hierarchical network is assumed where non-leaf sensor nodes aggregate data from their descendants. The neural networks predict the sensed data of a node N, provided the data reported by neighbors of the node N are given.

3 Intrusion Detection

Sensor nodes are vulnerable devices given by the nature of wireless communication and because of other limitations. They are usually deployed in open or even hostile environments where they can be easily manipulated or even stolen by an attacker. In order to keep their price low and to consume little energy, the nodes consist usually of a simple hardware where some conventional security countermeasures are unusable. Furthermore, a typical sensor node is not tamper resistant.

Attackers on WSNs can be categorized into two following classes [3, 16]:

1. *Passive attacker* uses his own device to listen on the ongoing communication to obtain sensitive data without any manipulation of the traffic within the WSN.
2. *Active attacker* uses his own device to disrupt the functionality of the WSN, manipulates the content of the ongoing packets, drops the packets, presents a fake identity to other sensor nodes or jams legitimate transmissions.

While passive attacks can be prevented by encryption, intrusion detection systems (IDSs) are used to detect some kinds of active attacks. In this chapter, we focus on *distributed* IDSs [1] where sensor nodes themselves monitor the overall network area by promiscuous listening on the transmissions among their neighbors.

We demonstrate the usability and benefits of IDS optimization using evolutionary computation for WSNs to detect *selective forwarding* and *delay* attacks. The description of these attacks and of our detection techniques proposed to detect these attacks follows.

3.1 Selective Forwarding Attack

Selective forwarding is one of the most widely discussed attack in WSNs [1, 3–7, 17]. When performing a selective forwarding attack, an attacker inserts a malicious sensor node into a WSN, where this node is believed as legitimate by the other (benign) sensor nodes. Once becoming a participant in packet routing, such a malicious sensor node can easily drop all the packets routed via itself (*blackhole* attack [3]) or can forward them selectively based on their contents, sensor measurements, source IDs or just randomly. Various attacker strategies are discussed in Sect. 5.

3.2 Delay Attack

The prerequisites for the delay attack are similar as for the selective forwarding attack—a malicious sensor node has to become a member of a routing tree in a WSN. Consequently, instead of dropping, the packets are intentionally delayed but finally forwarded. This kind of attack is aimed at time sensitive applications for WSNs where the delivery time of the packets to the BS is of a crucial importance. Such applications involve fire detection, people or animal movement detection and others. The same attacker strategies as for the selective forwarding attack can be applied for the delay attack.

3.3 Detection Techniques

We use *distributed* detection [1] where the IDS runs on each sensor node deployed in a WSN. Thus, the entire network area can be monitored to detect malicious behavior by sensor nodes themselves. However, this approach requires additional resources (e.g., memory and energy).

We provide two approaches of distributed detection:

1. *Non-collaborative detection*—no additional communication among the IDS nodes is required.
2. *Collaborative detection*—IDS nodes collaborate on the decision about monitored sensor nodes using exchange of *voting* packets.

We use the following notations to explain the functionality of the IDS [18]:

Notation 1 *The set* $A = \{a_1, \ldots, a_{n_m}\}$ *is a set of all malicious nodes in a network.*

Notation 2 *The set* $C = \{c_1, \ldots, c_{n_b}\}$ *is a set of all benign nodes in a network.*

Notation 3 *The function* $x : \mathbb{N} \to \mathbb{N}$ *takes a sensor node index as an argument, and returns a number of the neighbors that consider this node benign.*

Notation 4 *The function* $y : \mathbb{N} \rightarrow \mathbb{N}$ *takes a sensor node index as an argument, and returns a number of the neighbors that consider this node malicious.*

Notation 5 *The function* $n : \mathbb{N} \rightarrow \mathbb{N}$ *takes a sensor node index as an argument, and returns a number of the neighbors of this node.*

Notation 6 *The function* $m : \mathbb{N} \rightarrow \mathbb{N}$ *takes a sensor node index as an argument, and returns the amount of memory (in bytes) used by an IDS on this node.*

Neighbor $b_k \in C \cup A$ of a node $c_j \in C$ is each node such that c_j overheard at least one packet from b_k since the beginning of the WSN operation time.

Monitored neighbor $b_l \in C \cup A$ of a node $c_j \in C$ is such a *neighbor* of the node c_j that the IDS running on the node c_j collects the statistics of the packet forwarding of the node b_l. The selection process for the set of the monitored neighbors is described in Sect. 3.4.

Solution s is a specific configuration of the IDS in a form of a detection technique and specific values given to each of the parameters used by that technique.

Ranges of values of IDS parameters discussed in the following text are then discussed in more detail in Sect. 6.3.

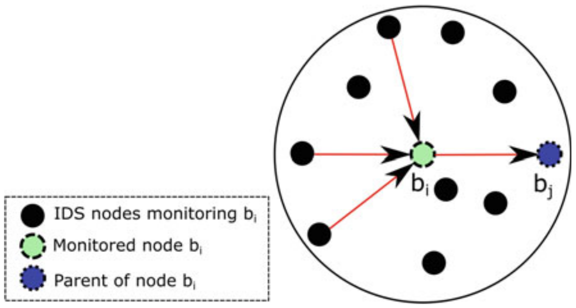
3.4 Non-collaborative Detection of Selective Forwarding Attack

In [12], we evaluated multi-objective evolutionary algorithms (MOEAs) on a simple IDS detecting selective forwarding attack. An IDS was running on each sensor node and continuously monitoring its own sent and also overheard packets addressed to all monitored sensor nodes whether they were forwarded or dropped by those monitored sensor nodes. Since the collaborative version used for the experiments is an extension of the non-collaborative version, we first describe the functionality of the non-collaborative version.

The basic principle is illustrated in Fig. 1. The black dots represent sensor nodes that are placed within communication range of sensor node $b_i \in C \cup A$ and, thus, can monitor b_i for selective forwarding attack. However, the number of monitored neighbors is limited to p_1 (*max monitored nodes*), not only due to memory reasons—the IDS can have incomplete information about furthest neighbors (the IDS nodes can be interfered, far from the monitored node or hidden behind an obstacle) causing additional false positives. Thus, each IDS monitors at most p_1 nearest neighbors (according to received signal strengths). The arrows represent routing directions of the packets—b_i forwards all received packets to a parent node $b_j \in C \cup A$. The IDS maintains a table, where each of p_1 rows corresponds to a certain monitored node. The table contains the number of packets received (PR) and forwarded (PF) by each monitored node.

The IDS stores all overheard packets addressed to all monitored neighbors in a single buffer limited to p_2 packets (*buffer size*). Each time a packet P addressed to

Fig. 1 Non-collaborative intrusion detection



a monitored node b_i is overheard by the IDS, the PR counter of b_i is incremented and packet P stored in the buffer. Once the node b_i forwards the packet P, the IDS increments PF of the node b_i and packet P is removed from the buffer. In case the packet P is being the oldest one and the buffer is full, it is removed from the buffer without incrementing the PF counter.

Finally, during the evaluation phase, a sensor node b_i is considered as attacker by the IDS node if two following conditions hold:

1. The IDS node has overheard (or sent) at least p_3 packets (*min received packets*) addressed to b_i.
2. The ratio of forwarded and received packets (*PF*/*PR*) is lower than p_4 (*detection threshold*).

Objective function 1 The number of *false negatives* (fn) of a solution s is calculated as follows:

$$fn(s) = \frac{1}{|A|} * \sum_{a_i \in A} \frac{x(a_i)}{n(a_i)}. \tag{1}$$

The values of fn range from 0 to 1. If every malicious node in the network is correctly detected by all its neighbors, fn is equal to 0 and if none of malicious nodes is detected by any of its neighbors, fn equals to 1.

Objective function 2 The number of *false positives* (fp) of a solution s is calculated as follows:

$$fp(s) = \frac{1}{|C|} * \sum_{c_i \in C} \frac{y(c_i)}{n(c_i)}. \tag{2}$$

The values of fp range from 0 to 1. If every benign node in the network is considered benign by all its neighbors, fp is equal to 0 and if all benign nodes are considered malicious by all its neighbors, fp equals to 1.

Objective function 3 The consumed *memory* (*mem*) in a solution s is calculated as follows:

$$mem(s) = 8 * p_1 + 16 * p_2, \tag{3}$$

8 bytes are required for every monitored neighbor (4 bytes for node ID, 2 bytes for PR counter and 2 bytes for PF counter) and 16 bytes are required for one slot in the buffer (4 bytes for source address, 4 bytes for receiver address, 4 bytes for destination address in a case of multiple base stations in the WSN and 4 bytes for unique ID of a packet). The memory demands come from our real security middleware ("WSNProtectLayer" for WSN [19]).

The values of mem range from 0, where the IDS is potentially switched off, to 288 bytes for our upper bounds of $p_1 = 30$ and $p_2 = 3$ used for the selective forwarding attack. For the delay attack, the upper bounds of $p_1 = 30$ and $p_2 = 10$ results in the maximum memory consumption of 400 bytes. The upper bound of p_1 (*max monitored nodes*) is based on our experiments. There is no significant improvement of any of the objectives with p_1 higher than 30. See Sect. 7 for more details. The upper bounds of p_2 (*buffer size*) are based on throughput analysis in the simulator.

3.5 Collaborative Detection of Selective Forwarding Attack

Collaborative detection of the selective forwarding attack that we first presented in [2] is an extended version of the non-collaborative detection discussed in Sect. 3.4. The idea behind the collaborative approach basically comes from [4] regarding to voting scheme and time windows. However, we enriched the collaborative approach presented in [4] by parameters "voting threshold" and "minimum received votes".

The monitored nodes are not evaluated by the IDS nodes at the end of the simulation. Instead, the simulation time is divided into windows of size p_5 (*time window*). The time windows are of the same fixed size among all the IDS nodes, but they are asynchronous—the first window of each IDS node is started randomly within the time interval of p_5. At the end of each time window, all monitored neighbors are evaluated by the IDS node and if an attack was detected, a voting process can be executed.

An example situation is depicted in Fig. 2. IDS node c_k monitors (among others) node b_i and detected too many packets dropped by b_i in a time window marked as "Attack!" in Fig. 3. This decision is based on the same principle as for the non-collaborative IDS discussed in Sect. 3.4. Since this time, IDS node c_k considers a node b_i malicious "locally"—still no alert is produced.

Then, the decision of the node c_k is checked with all neighbors of c_k. Thus, c_k broadcasts a voting request to its neighbors (arrows from c_k in Fig. 2 point the neighbors of c_k that can also monitor node b_i). Each of the asked nodes that also monitors node b_i answers at the end of its own time window. If an asked IDS node consider b_i an attacker (either just locally or globally), it answers positively, otherwise negatively. Node c_k waits the following time window to collect the responses. Finally, the monitored node b_i is considered an attacker "globally" by c_k and c_k produces an alert if two following conditions hold:

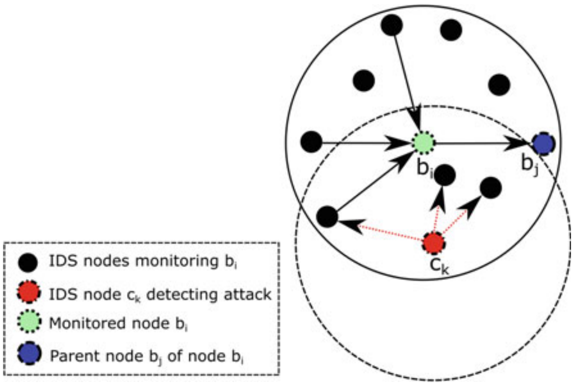


Fig. 2 Collaborative intrusion detection

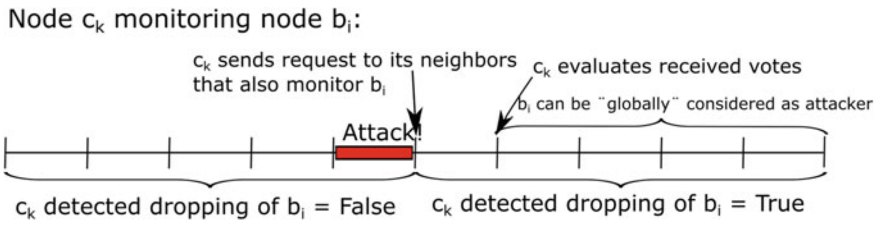

Node c_k monitoring node b_i:
c_k sends request to its neighbors that also monitor b_i
c_k evaluates received votes
b_i can be "globally" considered as attacker
Attack!
c_k detected dropping of b_i = False
c_k detected dropping of b_i = True

Fig. 3 Time windows. Since the end of the window in which the attack was detected, node c_k always votes positively about node b_i. Node b_i can be globally considered malicious by c_k since the end of window following the one in which the attack was detected if the voting result is positive

(i) At least p_6 votes (*min votes received*) were received.
(ii) The ratio of positive and all responses is at least p_7 (*voting threshold*).

Malicious nodes can falsely report to the IDS nodes to defend another malicious node. We consider this to be an specific attacker strategy that is discussed in Sect. 5.

Objectives We use the same three objectives as in Sect. 3.4.

3.6 Collaborative Detection of Delay Attack

Time related attacks result in long delays and traffic imbalance [20]. We believe that a WSN should guarantee the delivery time for some applications (e.g., movement monitoring or fire detection). In [1], the *delay detection rule* is defined as follows: "The transmission of a message by a monitor's neighbor must occur before a *defined timeout*". We adapt this rule to our IDS detecting delay attack.

A technique that is similar to that one used for the selective forwarding attack can be extended to detect intentional delays. Using the buffering technique discussed above, a packet can finally be considered forwarded even though some malicious node delayed its transmission. If the size of the buffer is not exhausted, the monitored packet can be stacked in the buffer for a long time and finally forwarded by the monitored neighbor with a big delay before being removed from the buffer by the IDS. Thus, such a packet is undetected for selective forwarding attack, yet useless for a base station if real-time sensing is required.

An important issue to consider is how long the IDS should wait until the packet is considered delayed and how many packets have to be delayed to consider a monitored neighbor as a *delay* attacker. We suggest to assign a *time attribute* to each of the buffered packets. If a predefined timeout passes, the packet is considered delayed. As for the selective forwarding attack, an alert is produced when p_5 time units pass and the ratio of delayed packets is higher than p_4. In such a case, the *majority voting scheme* is applied for a decision about the *delay attack*.

Our detection technique of the delay attack was proposed in [2] and is evaluated thoroughly in this chapter. The detection technique extends the selective forwarding attack detection specified in Sects. 3.4 and 3.5. As we focus on collaborative detection in this chapter, we do not consider non-collaborative detection of the delay attack. We incorporate another parameter p_8 (*delay timeout*) that is a timeout when a packet in the IDS buffer is marked as delayed.

Objectives We use the same three objectives as in Sect. 3.4.

4 Computational Intelligence-Based Optimization

In this section, we present our optimization framework that was designed in [18]. The optimization framework consists of two main components: a *network simulator* and an *optimization engine*. The simulator is used for evaluation of the candidate solutions (in a form of IDS configurations) that were designed by the optimization engine. Based on the simulation results in a form of metrics described in Sect. 3.4 (false positives, false negatives, memory), the optimization engine produces a new generation of solutions.

We enhance the optimization framework to easily distribute the computation of the individuals in each generation to multiple computers using BOINC (Berkeley Open Infrastructure for Network Computing) distributed computing platform [21] and to use MOEAs. The efficiency of MOEAs for the problem of IDS optimization for the WSNs was evaluated in [12]. Also the MOEAs configuration issue was addressed there. The findings are utilized in this work.

4.1 Simulator

We use the MiXiM simulator [22] that is based on OMNeT++ simulation platform [23]. The selection of the simulator is based on our previous results on the comparison of various simulators with the reality [24]. MiXiM is a discrete event simulator with a good support of wireless channel modeling and support of all communication layers of the current sensor nodes.

However, as we showed in a previous work [24], a proper calibration of all the models is required to obtain sufficiently accurate results. The wireless channel model is based on a *log normal shadowing* [25] that is widely used for wireless communication modeling. Two parameters of the model has to be set up according to modeled environment. The values of the parameters can be either based on measurements of wireless signal propagation in the target area or based on recommended values for target environment type [25]. For experiments in this chapter, we use recommended values for an outdoor environment (path loss exponent equals to 2 and standard deviation of the attenuation equals to 2). All simulated sensor nodes transmit packets with power equals to -25 dBm (transmission level 3 of TelosB [26] sensor nodes). See [24] for more information about our calibration approach or [25] for the theoretical background.

4.2 Evolutionary Algorithms

We found evolutionary algorithms to be a very efficient metaheuristic solving optimization of the IDS for WSNs. As we showed in [12], optimal or near-optimal results can be found in a feasible time. For a problem solved in this chapter, we are not able to compute exhaustive search even with the computational grid we have at our disposal. Thus, we compare the results found by evolution with much more time demanding sampling.

In our early work [18], we used *single-objective* evolutionary algorithms, where a fitness function blending all three objectives with user-specified weights had to be defined. The main disadvantage was the amount of required experience in weight definition. If the network operator wanted to change the weights, the optimization process had to run again. Thus, based on [12], we recommend to use *multi-objective* evolutionary algorithms (MOEAs) that eliminate the mentioned problem.

Using MOEAs, the network operator can choose any IDS setting from the *Pareto front*[1] [27] approximation and change the selection to another optimized one any time according to current requirements. Based on results from [12], where we evaluated 48 different MOEA's configurations for two widely used algorithms—NSGA-II [28] and SPEA2 [29], we configure the evolution as follows in this chapter: The population consists of 200 individuals, 200 generations are computed, probability of multi-point crossover is 0.5 for all experiments. Mutation probability of each parameter is 0.01 or 0.25 for results marked as "Evo #1" and "Evo #2", respectively. If mutation is

[1]Pareto front is a set of non-dominated solutions with respect to all objectives. Thus, a network operator can easily choose between a solution A with a better IDS accuracy but higher resource consumption or solution B with a worse IDS accuracy but lower resource consumption. Solution C, that is dominated by A and B in all objectives is dominated and, thus, is not a member of the Pareto front.

performed, the value is shifted randomly within the interval of 10 % of the overall parameter range. NSGA-II is used in all experiments.

Performance Metrics Since we are not able to compare the Pareto front approximations found either by sampling or by evolution with the true Pareto front, we use two metrics to mutually compare the different optimization strategies. These metrics are also used to compare the IDS performances across different attacker strategies.

1. *Hyper-volume indicator* [27, 30]—Hyper-volume indicator is given by the calculation of a volume of the objective space that is dominated by evaluated Pareto front approximation. A reference point R that is dominated by each solution in the Pareto front approximation has to be established—as an upper bound of each of the objectives. The reference point serves as an upper bound of the dominated objective space for the volume calculation. Minimal values of the coordinates of the point R are maximal values of each of the objectives across the population of the evaluated Pareto front approximation.
   In all our experiments where the hyper-volume indicator is calculated, we normalize the objective function *memory* dividing its output by its upper bound (288 for the selective forwarding attack and 400 for the delay attack). The results of the normalized *memory* function range from 0 to 1. The value of the reference point is established to: $R = [1, 1, 1]$ for all calculations of the hyper-volume indicator in this work to enable mutual comparisons.
   The value of the hyper-volume indicator ranges between 0 (potential unrealistic worst single solution of the IDS setting in the Pareto front approximation consuming all possible memory in spite of evincing the values of false positives and false negatives equal to 1) and 1 (potential unrealistic ideal single solution of the IDS setting in the Pareto front approximation consuming no memory and evincing the values of false positives and false negatives equal to 0).
   We use the software presented in [31] to calculate the hypervolume-indicator.
2. *Coverage metric* [32]—Coverage metric enables mutual comparison of two different Pareto front approximations. It is used to calculate the percentage of solutions found in the Pareto front approximation A that are not dominated by any solution found in the Pareto front approximation B, and vice versa.
   The value of the coverage metric when calculating percentage of solutions in A dominated by solutions in B ranges from 0 % when none of the solutions in A is dominated by any solution in B to 100 % when each solution in A is dominated by at least one solution in B. Note that if a mutual comparison is required, we also have to calculate the percentage of solutions in B dominated by solutions in A resulting in two numbers all together as the coverage metric output.
3. *Number of simulations*—Since the evaluation of each individual by simulator is time demanding (approx. 5–8 min), we also provide a number of simulation runs needed within the whole optimization process.
4. *Number of non-dominated solutions*—This metric denotes the number of non-dominated solutions in the resulting Pareto front approximation.

5 Attacker Strategies

The aim of this section is to elaborate on the strategies an attacker can take to efficiently selectively forward or drop the packets. We also discuss the impact of each of the strategy on the IDS performance. Furthermore, we provide a set of recommendations with respect to corresponding attacker strategies for a network operator before setting up the IDS optimization process.

We divide the concepts of attacker strategies into the three following categories:

1. *Attacker behavior*—content-based selection, ratio of dropped and all received packets by a malicious node, malicious voting.
2. *Deployment of malicious nodes*—strategy of malicious sensor nodes deployment in a WSN.
3. *Number of deployed malicious nodes*—ratio of malicious and all sensor nodes in a WSN.

Note that if only the selective forwarding attack is discussed in the following text, the attacker strategy for the delay attack would be equivalent. The only difference would be that selected packets on malicious nodes are delayed instead of being dropped.

5.1 Attacker Behavior

A malicious sensor node can take various strategies w.r.t. the decision whether a packet should be forwarded or not. In the ultimate case, all the packets can be dropped by a malicious node $a_i \in A$, effectively changing the selective forwarding attack to a *blackhole* attack [3]. However, the blackhole attack can be easily detected either by our distributed IDS running on the neighboring sensor nodes (dropping ratio higher than any value of the *detection threshold* p_4 is detected). Furthermore, the base station not receiving any packets from such sensor node a_i and also from all the descendants of the a_i can easily detect the attack.

We discuss different approaches of the selection of the packets to be dropped that an attacker can use to decrease the probability of being detected. Furthermore, we discuss impacts of malicious voting.

Dropping Ratio Random dropping with a given ratio is probably the most simple way of selective forwarding attack. The attacker specifies the percentage s_1 of dropped packets by a malicious node $a_i \in A$ as the only parameter for the random dropping. Consequently, the packets that should be forwarded by a_i are dropped with the probability s_1.

Recommendation for the IDS We recommend usage of the collaborative version of the IDS optimized for a target WSN as discussed in Sect. 3. A network operator decides carefully on the minimum percentage s_1 of the packets dropped by the sensor nodes that is considered unacceptable. Such a dropping rate is to be set up for all the simulated malicious sensor nodes. The optimization process adapts all the IDS parameters, particularly the parameter p_4 (*detection threshold*), taking into account also packet losses caused by interference, congestion and other aspects of the unreliable wireless communication. When such an optimized IDS is used in a real WSN and any malicious (or malfunctioning) sensor node drops packets with even a higher frequency, then such sensor node is detected even more reliably. In the other case, if the dropping ratio of a monitored node is lower, the selective forwarding attack may not be detected because such a behavior is considered acceptable.

Dropping in Bulk Another behavior of a malicious node $a_i \in A$ can be dropping the packets in a bulk. It means that all or some percentage s_1 of the packets can be dropped by a_i, but only within specific time intervals with a minimum length of s_2 seconds for each of them. This strategy can be used to suppress transmitted packets during some specific sensitive period (e.g., dropping packets while a physical attacker is approaching the area of a WSN that detects people movement and protects some environment against unauthorized access).

Recommendation for the IDS We recommend to set up the parameter p_5 (*time window*) of the IDS to the length of $\frac{s_2}{2}$ at maximum for the minimum length of the interval s_2 seconds that is considered unacceptable. This countermeasure ensures that all the IDSs in the neighborhood of the malicious sensor node a_i have a chance to detect dropping (if it is higher than p_4—*detection threshold*) within a single time window—not decreased by a non-attacking phase of the malicious node a_i. If such an upper bound for the parameter p_5 is set up during the optimization, the other IDS parameters are optimized accordingly—particularly the parameter p_3 (*minimum received packets*) depends on the length of the time window. Note that both recommendations for the *dropping ratio* and the *dropping in bulk* complement each other.

Content Based Dropping A malicious node $a_i \in A$ can drop the packets based on their contents distinguishing important and non-important packets. To give an example of such a situation, the important packets can inform the base station about the presence of an intruder while the non-important packets can be periodically sent "still-alive" packets informing the base station that the sensor nodes are in an operation mode.

Recommendation for the IDS We recommend to extend the IDS by a separate buffer and table for such important packets—a monitoring node will separately monitor important and non important packets. This would mean additional requirements on the amount of memory consumed by the IDS (approx. doubled). However, the optimization process is analogous for both types of the packets. The definition of the malicious behavior in the context of the *dropping ratio* and the *dropping in bulk* discussed above is left to the network operator for both types of the packets.

Source Based Dropping A malicious node $a_i \in A$ can drop the packets based on their source addresses in order to drop packets from, e.g., some specific location.

Recommendation for the IDS In some cases, we believe that such a selective dropping based on the source address can be detected without any changes of the IDS functionality. However, the network operator should consider the impact of the selective forwarding based on the source address on the overall percentage of dropped packets by the monitored sensor node. Maintaining separated IDS buffers and tables for each source address can increase the sensitivity of the IDS on such source based dropping at the cost of increased memory consumption.

Delay Interval When a delay attack is executed, a malicious node $a_i \in A$ can delay the packets (by any strategy discussed above) for a fixed timeout d_1 seconds or randomly within interval $\langle d_2, d_3 \rangle$ seconds.

Recommendation for the IDS A network operator should decide carefully on the minimum timeout d_1 of packets delayed by the sensor nodes that is considered unacceptable. Such a delay timeout is to be set up for all the simulated malicious sensor nodes. The optimization process adapts all the IDS parameters, particularly the parameter p_4 (*detection threshold*) and the parameter p_8 (*delay timeout*), taking into account also packet losses caused by interference, congestion and other aspects of unreliable wireless communication. When such an optimized IDS is used in a real WSN and any malicious (or malfunctioning) sensor node delays packets with even a higher frequency or a longer timeout, then such sensor node is detected even more reliably. In the other case, if the dropping ratio of a monitored node is lower or the delay timeout is shorter, the delay attack may not be detected because such a behavior is considered acceptable.

Malicious Voting A malicious node(s) can falsely vote for benignity in order to defend another malicious node.

Recommendation for the IDS Considering such attack on the IDS, a network operator should adjust p_7 (*voting threshold*). For example, if it is assumed that up to 20 % of neighbors can be malicious, the *voting threshold* should be lower than 0.8 because up to 20 % neighbors vote falsely for benignity. The precise value of the *voting threshold* is subject of the optimization and consequently preferences—there is a trade-off between false positives and false negatives.

5.2 Deployment of Malicious Nodes

An attacker can deploy malicious sensor nodes into a WSN in specific "patterns" [33, 34]. However, in most papers, the deployment strategy of the sensor nodes into a WSN is left unresolved [5, 17] or the placement of malicious sensor nodes is selected randomly [4, 6, 7].

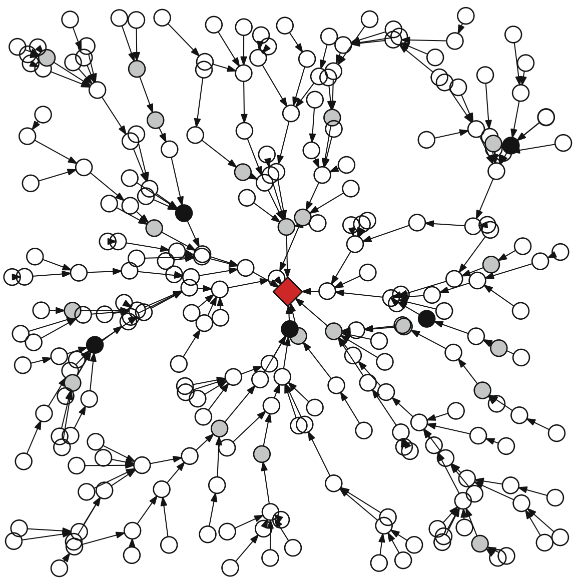


Fig. 4 Topology of the evaluated WSN for the random attacker strategy. The sensor nodes are represented by *circles* while the base station is represented by the *red diamond*. The *black circles* represent malicious sensor nodes for the scenario with 2 % malicious sensor nodes and together with the *gray circles* for the scenario with 10 % malicious sensor nodes (color figure online)

Influence of several realistic malicious nodes deployment strategies on the IDS performance is discussed in this subsection. It might be impossible for the network operator to predict the attacker deployment strategy. Thus, the impact of all discussed attacker strategies on the IDS parametrization is evaluated in Sect. 7 to give a clue how the malicious sensor nodes can be deployed in the simulations to obtain as robust IDS parameters as possible.

Random Attacker Strategy The random attacker strategy is the most widely considered strategy by IDSs for WSNs [4, 6, 7]. In this approach, the malicious sensor nodes are inserted into the WSN on random positions. However, we assume this attacker strategy being far from the behavior of a real attacker in most cases. An attacker can have access to or may be interested in only a specific part of the environment. However, this attacker strategy can be utilized for IDS optimization. If a sufficient number of sensor nodes is considered, the random deployment can cover different places (close or far from the base station, with sparsely or densely deployed sensor nodes, etc.) at the same time.

We parameterize this attacker strategy by a percentage of nodes controlled by an attacker by s_3. In Fig. 4, we give an illustration of the random attacker strategy that we evaluate in Sect. 7.

Center Drop Attacker Strategy The goal of an attacker within this strategy is to compromise sensor nodes surrounding the base station. Alternatively, an attacker can choose an arbitrary target place in the WSN and compromise its surrounding sensor nodes.

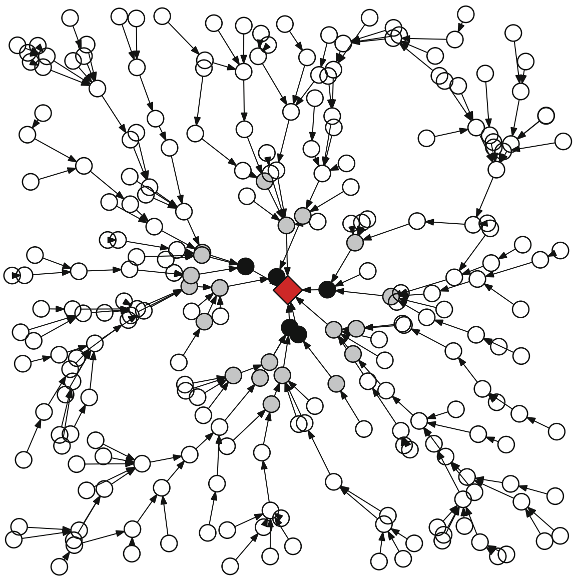


Fig. 5 Topology of the evaluated WSN for the center drop attacker strategy. The sensor nodes are represented by *circles* while the base station is represented by the *red diamond*. The *black circles* represent malicious sensor nodes for the scenario with 2 % malicious sensor nodes and together with the *gray circles* for the scenario with 10 % malicious sensor nodes (color figure online)

In this chapter, we evaluate a situation where the sensor nodes surrounding the base station are malicious. We parameterize this attacker strategy by s_4—percentage of the malicious sensor nodes in the WSN ordered by a distance from the base station. In Fig. 5, we give an illustration of the random topology that we evaluate in Sect. 7.

Direct Center Attacker Strategy In the direct centre attacker strategy, we consider an attacker passing through the WSN along a line segment, reaching the base station and leaving the WSN on the opposite direction. We assume an attacker can compromise the sensor nodes located nearby his or her trajectory.

The percentage sensor nodes ordered by distance to the trajectory is parameterized by s_5. Our test case and an example of an attacker passing through the WSN from left side to the right side through the base station is depicted in Fig. 6.

5.3 Number of Deployed Malicious Nodes

The number of malicious sensor nodes in the WSN can vary—based on the options of an attacker—for each deployment strategy. In fact, parameters s_3–s_5 reflect the percentage of deployed malicious sensor nodes for each of the deployment strategy. We analyze the influence of the changed percentage of present malicious sensor nodes from the situation during the optimization in Sect. 7.

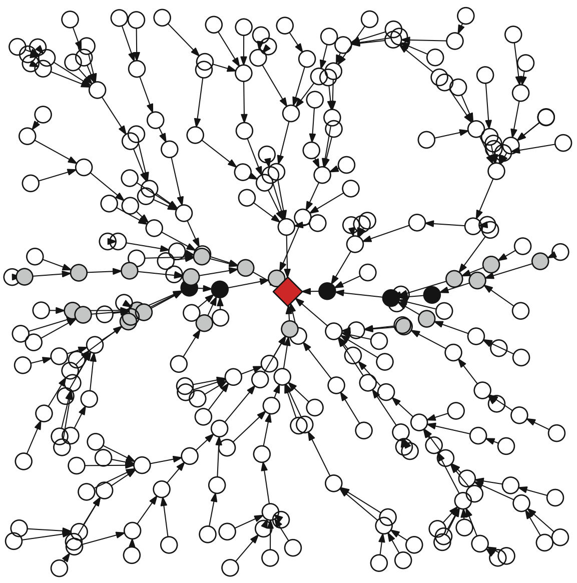


Fig. 6 Topology of the evaluated WSN for the direct center attacker strategy. The sensor nodes are represented by *circles* while the base station is represented by the *red diamond*. The *black circles* represent malicious sensor nodes for the scenario with 2 % malicious sensor nodes and together with the *gray circles* for the scenario with 10 % malicious sensor nodes

5.4 Robustness Evaluation of Optimized Solutions on Different Attacker Strategies

One of the contributions of this chapter is an optimization of the IDS parameters for different attacker strategies. The problem is that the concrete malicious node deployment cannot be reliably predicted in advance. In order to discuss the robustness of the found optimized solutions, we evaluate the performance of optimized IDS for a given attacker strategy against other attacker strategies in Sect. 7. We use two approaches of the evaluation of the impact of changes in the attacker strategy on the IDS performance, the description of which follows.

We use the following notation to explain the evaluations we performed to compare IDSs optimized for various attacker strategies:

Notation 7 *The set* $AS = \{as_1, \ldots, as_{n_{as}}\}$ *is a set of all* n_{as} *attacker strategies evaluated in this chapter.*

Notation 8 *The set* $PF = \{pf_1, \ldots, pf_{n_{as}}\}$ *is a set of all Pareto front approximations. Each Pareto front approximation* $pf_i \in PF$ *where* $1 \leq i \leq n_{as}$ *was optimized for an attacker strategy* as_i*.*

Single Pareto Front Approximation Evaluation on Multiple Attacker Strategies For each Pareto front approximation $pf_i \in PF$ optimized for an attacker strategy as_i, we compute the performance of each IDS setting from pf_i for all other attacker strategies $as_j \in AS \setminus \{as_i\}$. This way, we evaluate the changes of the IDSs performances optimized for a specific attacker strategy in a situation where another attacker strategy operates.

Multiple Pareto Front Approximations Evaluation on Single Attacker Strategy For each attacker strategy $as_i \in AS$, we compare the performances of the IDS settings in all Pareto front approximations $pf_j \in PF \setminus \{pf_i\}$ with the performances of the IDS settings in Pareto front approximation pf_i. This way, we evaluate how the IDSs performs in a situation where another strategy operates comparing to the IDSs that were optimized for that situation.

6 Experiment Settings

In this section, we describe experiment settings and optimization scenarios that we use for evaluation of our IDSs. We also present the ranges of IDS parameters.

6.1 Application

The simulated WSN that we evaluate in this work is inspired by the police unit scenario in [19]. Each sensor node sends "still alive" packets every second. These packets can be either dropped or delayed by malicious sensor nodes. The main goal of our optimization framework is to optimize the IDS for a given scenario (application, topology, environment, etc.) and to be robust for various attacker strategies in that environment. We do not aim to provide general IDS setting for any WSN.

One hour of the WSN operation time is simulated in all evaluations.

6.2 Topology and Routing

We build on the topology and routing same as in [12], so that we are able to compare the collaborative and non-collaborative IDS results. The network consists of 250 uniformly distributed sensor nodes deployed in an area of 200 m $\times$ 200 m. The average area for one node is 160 m^2 and the distance between two nearest neighbors is 12.65 m on average. During the simulation, a node $b_j \in C \cup A$ has 41 neighbors (nodes from which b_j heard at least one packet during the simulation) on average.

The routing tree is static with longest branches of 8 hops. The topology and the routing tree are depicted in Figs. 4, 5 and 6.

6.3 IDS Parameters and Sampling

In Table 1, we summarize all eight parameters of the IDS presented in Sects. 3.4–3.6, their maximum and minimum values. Steps and sampling values are used for a time demanding sampling that we computed to compare the results found by the evolution.

Table 1 The list of IDS parameters

Name	Description	Range	Step	Sampling
p1	*Maximum monitored nodes*	⟨1, 30⟩	1/3	3, 9, 27
p2	*Buffer size*	⟨1, 3⟩/⟨1, 10⟩	1	1, 2, 3/3, 6, 9
p3	*Minimum received packets*	⟨1, 30⟩	1/5	1, 15, 30
p4	*Detection threshold*	⟨0.05, 0.95⟩/⟨0.1, 0.9⟩	0.05/0.1	0.25, 0.5, 0.75
p5	*Time window*	⟨10, 300⟩	10/30	10, 150, 300
p6	*Minimum received votes*	⟨1, 10⟩	1/2	1, 5, 10
p7	*Voting threshold*	⟨0.1, 1⟩	0.1	0.25, 0.5, 0.75
p8	*Delay timeout*	⟨1, 5⟩	1	1, 3, 5

If multiple values are presented and divided by "/", the first values were used for the detection of the selective forwarding attack and the second values then for the delay attack detection

Sampling In order to show that evolution can find good enough results in reasonable time, we compared the results found by MOEAs with a true Pareto front found using exhaustive search on multiple computers in [12]. However, we are not able to compute all possible settings for this more complex IDS with additional parameters even if we can run about 200 simulations in parallel in our computational cluster. The exhaustive search would require 148,770,000 simulation runs for the scenario with the selective forwarding attack and 2,479,500,000 for the scenario with the delay attack if all possible settings would be evaluated. One simulation takes approx. 5–8 min.

We decided to sample the search space in the following way. For each parameter p_i, where $i \in \{1, \ldots, 7\}$ for the selective forwarding attack and $i \in \{1, \ldots, 8\}$ for the delay attack, we choose three carefully considered "sampling" values presented in Table 1. The selection is based on experience and results obtained during early experiments. Then, we iterate over all parameters $p_1, \ldots, p_7$ for the selective forwarding attack, respective $p_1, \ldots, p_8$ for the delay attack. For each of the parameters, we evaluate all settings within their ranges using steps provided in Table 1. For each value of each parameter p_i, we evaluate all "sampling" settings of all other parameters. Using this approach, we reduce the number of simulations to 84,564 for the selective forwarding attack and to 122,472 for the delay attack.

Having the set of solutions obtained from the aforementioned sampling, we extract only those solutions that are not strictly dominated by any other solutions within this set. We call this extracted set the *Sampling Pareto front approximation*. This set is compared to Pareto front approximations found by evolutions. Note that finding the *Sampling Pareto front approximation* is much more computationally demanding than finding Pareto front approximations using evolution as discussed in Sect. 7. We computed the sampling for an attacker strategy with randomly deployed 5 (2 %) malicious sensor nodes for both selective forwarding and delay attacks.

6.4 Evaluated Attacker Strategies

In all experiments, the malicious sensor nodes drop randomly 50 % of packets that should be forwarded for the selective forwarding attack. For evaluation of the delay attack, the malicious sensor nodes delay all packets within interval of $\langle 0, 5\rangle$ s.

Deployment of Malicious Nodes We optimize the IDS for six different deployments of malicious sensor nodes for the selective forwarding attack. For each deployment strategy (random, center drop and direct centre), we evaluate two cases: with 5 malicious sensor nodes (2 %) and with 25 malicious sensor nodes (10 %). All attacker strategies are illustrated in Figs. 4, 5 and 6.

For the delay attack, we evaluate only the case with 5 (2 %) randomly deployed malicious sensor nodes due to computational time restrictions. However, the characteristics that are related to overhearing the communication and to the collaborative decision are equivalent to the selective forwarding attack that we evaluated thoroughly.

No "leaf" sensor node (a node that has no descendant) is malicious within the experiments. These sensor nodes can neither perform efficiently the selective forwarding or the delay attack (no packets are forwarded by them), nor can be detected by any IDS node (no packets addressed to them can be overheard).

7 Experiment Results

In this section, all experiment results are presented and discussed. First, we show the increased accuracy of the collaborative IDS in comparison with the non-collaborative IDS, in the case of detecting the selective forwarding attack. Then we compare evolution performance for selective forwarding and delay attacks with much more time demanding sampling. Finally, we provide a mutual comparison of IDS settings optimized for various attacker strategies.

7.1 Selective Forwarding Attack

First, we briefly compare the Pareto front approximation found by sampling for the collaborative IDS with a true Pareto front found for the non-collaborative IDS in [12]. Both detection techniques are evaluated in the same WSN and attacker strategy—random deployment of 2 % malicious sensor nodes. In Fig. 7, we show different views

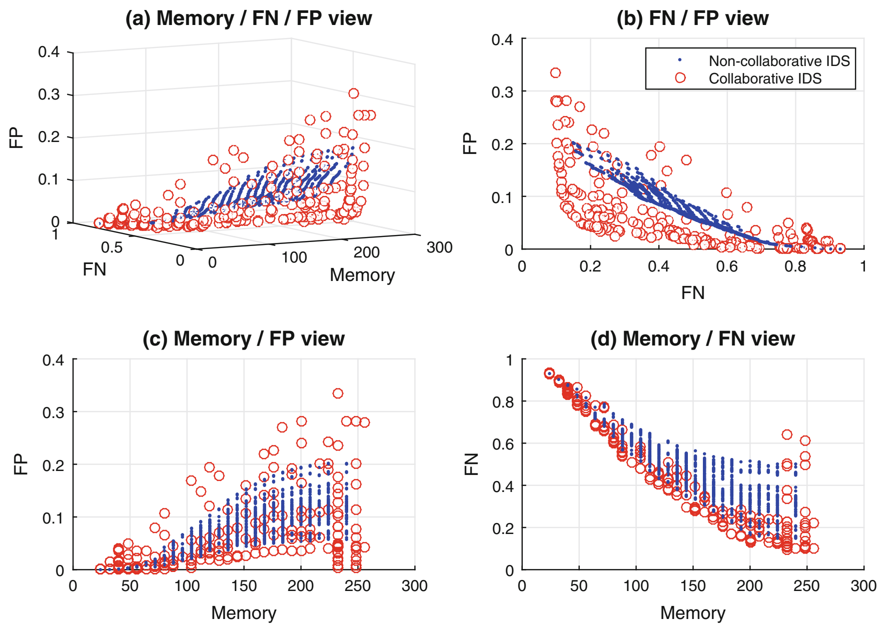


Fig. 7 Sampling Pareto front approximation for the collaborative selective forwarding attack detection compared to the true Pareto front for the non-collaborative selective forwarding attack detection. All the solutions found by sampling for the collaborative detection dominate the solutions found by exhaustive search for the non-collaborative detection

of the optimized solutions for both detection techniques in the three-dimensional objective space.

In Fig. 7a, we show that all sampled non-dominated solutions found for collaborative IDS dominate the Pareto optimal solutions found for non-collaborative IDS. Measuring the dominated volume of the objective space by the hyper-volume indicator, we obtain 0.525 for the non-collaborative IDS and 0.574[2] for the collaborative IDS. In Fig. 7b, we show that the collaboration among the IDS nodes can significantly decrease the number of false positives—a consensus has to be made to label a node as attacker. A decrease of false negatives is caused by dividing the monitoring time into smaller windows, where, in each of them, a potential dropping can be detected. In Fig. 7c, d, we can see that higher memory consumption caused particularly by a higher number of monitored neighbors decreases false negatives on one hand (more neighbors being monitored means also higher number of truly recognized malicious nodes), but increases false positives on the other hand (if a neighbor is not monitored, it can neither be labeled as malicious one truly, nor falsely).

As we shown, the collaborative approach provides better IDS results even though we are not able to evaluate the whole search space. However, the collaborative IDS

[2]As shown below, evolution can improve the results farther.

requires a communication overhead. Nevertheless, we would like to emphasize that the overhead is not significant—at least for our simulated application. Each sensor node $c_k \in C$ initiates the voting scheme at most once for each monitored neighbor $b_l \in A \cup C$ during the whole WSN operation time. Each neighbor $c_i \in C$ of the c_k has to answer to the voting request if the b_l is also monitored by the c_i. Sensor nodes monitor 30 neighbors at most and are monitored by at most 30 neighbors on average. That means that c_k initiates 30 voting request at most in case all monitored neighbors are suspicious. The average sensor node c_k has to answer on 30 voting requests (but only in the unrealistic worst case—only if it also monitors the sensor node in the request) on average for each of 41 neighbors (see Sect. 6.2 for more details). The overall overhead would be 1260 packets sent by each IDS node in such very unrealistic worst case. Note that the sensor nodes in a distance of one hop from the base station forward approx. 2500 packets during only one hour of the operation time of the WSN.

Evolution can speedup the process of finding solutions that are similar or even dominate the solutions found by the sampling. We present results of two multiobjective evolution runs (marked as "Evo #1" and "Evo #2"). The evolution settings are described in Sect. 4.2. The solutions found by the evolution compared to those found by sampling are depicted in Fig. 8.

In Table 2, we present results of all four metrics for each of the optimization process. We can see that both evolution runs outperforms the sampling according to

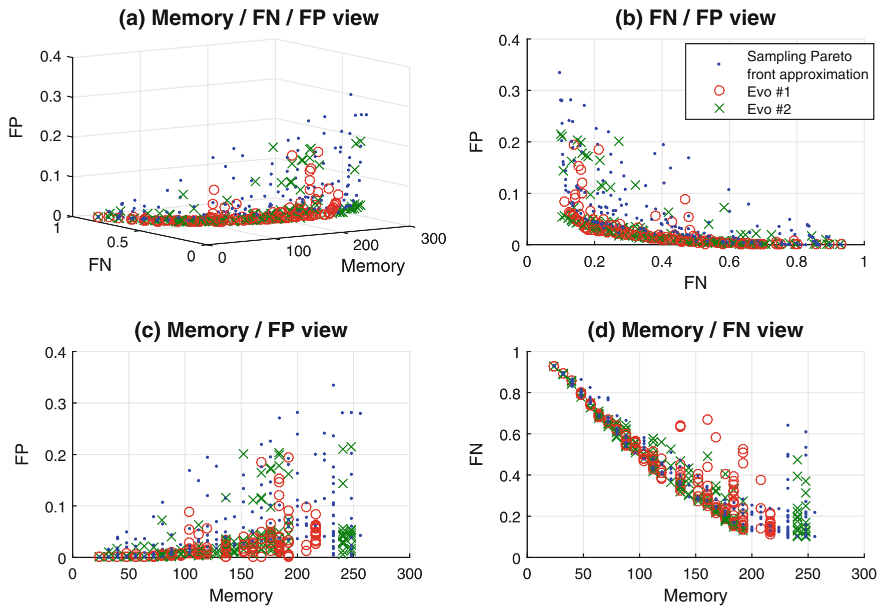


Fig. 8 Results for detection of the selective forwarding attack found by evolution compared to the results of the Sampling Pareto front approximation

Table 2 Performance metrics of the optimization of the selective forwarding attack

	Hyper-volume	Coverage			Simulations	Solutions
		Evo #1	Evo #2	Sampling		
Evo #1	0.577	–	65 % (94/144)	100 % (144/144)	13,502	144
Evo #2	0.583	82 % (126/153)	–	98 % (150/153)	23,558	153
Sampling	0.574	22 % (45/201)	19 % (39/201)	–	84,564	201

Hyper-volume indicator—result of the hyper-volume indicator for each optimization process. *Coverage metric*—for each Pareto front approximation in a row, the values specify the number of found solutions that are not dominated by any solution within the result set of Pareto front approximation specified in the column. *Number of simulations*—number of simulation runs within the whole optimization process. *Number of non-dominated solutions*—number of resulting non-dominated solutions

the hyper-volume indicator. Measuring the performance by mutual coverage of the solutions, we can see that solutions found by the evolution runs dominate nearly all solutions found by the sampling. However, we were able to find several solutions that have a lower number of false negatives using the sampling than any solution found by the evolution (see Fig. 8b). The process of optimization by the evolution was less time demanding as declare the numbers of required simulations. More non-dominated solutions were found by the sampling. We recommend to use a larger population size to obtain higher number of non-dominated solutions, if needed. However, since the solutions are well spread through the objective space (see Fig. 8), we do not consider the lower number of found non-dominated solutions as an important disadvantage.

IDS Parameters Discussion We discuss the IDS parameters of the solutions found by “Evo #2” as its Pareto front approximation evinces best performance.

In the resulting set of solutions, we can find nearly all possible settings of the p_1 (*max monitored nodes*) equally distributed. The values absolutely correlate with the objective function 3—memory consumption, since all solutions found use a *buffer size* (p_2) equaled to 1. No other parameter influences the memory consumption. Thus, the impact of the maximum number of monitored neighbors on the IDS performance can be directly observed in Fig. 8 (the axis “Memory”). W.r.t. the *buffer size*—we can find several solutions having the buffer size equaled to 2 in all attacker strategies except for the random one (see Sect. 7.3 for other evaluated strategies). Some of the malicious sensor nodes requires bigger buffer size due to being close to the BS encountering higher traffic.

The values of the *min received packets* (p_3) varies between 1 and 11 (5.07 on average). The values of the *detection threshold* (p_4) varies between 0.45 and 0.55 (0.504 on average), the *time window* (p_5) between 43 and 294 s (210 s on average), the *minimum received votes* (p_6) between 1 and 7 (2.79 on average) and the voting threshold between 0.28 and 0.99 (0.86 on average).

7.2 Delay Attack

In this section, we present non-dominated results for the detection of the delay attack found both by the sampling and evolution. In Fig. 9, we compare the performance of the IDS settings found by both the evolution and the sampling in the objective space. The number of non-dominated solutions found by sampling is reduced comparing to the selective forwarding attack. It may be caused by the fact that we are not able to compute such "dense" sampling—the search space is much larger (see Table 1). Since the basic principle of the delay attack detection is similar to the selective forwarding, we can observe similar patterns of the Pareto front approximations. Main difference is higher memory consumption needed to obtain comparable false negatives. This is caused by a need of storing the packets in the IDS buffer for longer time.

In Table 3, we can see that the evolution provides better results than the sampling w.r.t all the metrics—similarly to the selective forwarding attack.

IDS Parameters Discussion Such as for the selective forwarding attack, results found by "Evo #2" are discussed for the delay attack.

The number of *max monitored nodes* (p_1) varies between 1 and 24 (15.9 on average). The *buffer size* (p_2) varies between 1 and 9 (2.3 on average) and only the solutions with p_1 higher than 18 requires more than 3 slots in the buffer. The parameter *min received packets* p_3 evinces values between 1 and 8 (2.2 on average). The values of the *detection threshold* (p_4) varies between 0.07 and 0.64 (0.38 on average).

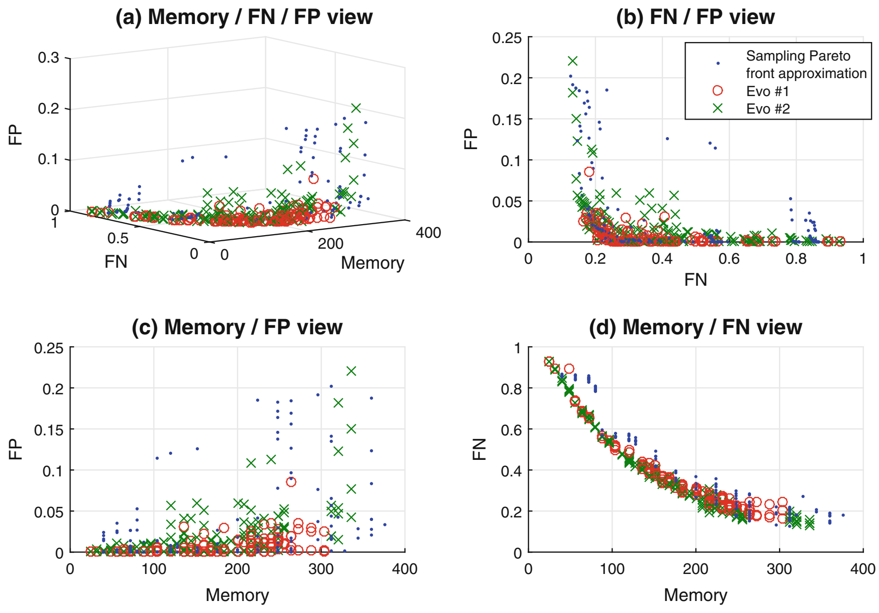


Fig. 9 Results for detection of the delay attack found by evolution compared to the results of the Sampling Pareto front approximation

Table 3 Performance metrics of the optimization of the delay attack

	Hyper-volume	Coverage			Simulations	Solutions
		Evo #1	Evo #2	Sampling		
Evo #1	0.604	–	61 % (72/118)	97 % (115/118)	13,539	118
Evo #2	0.625	79 % (112/141)	–	99 % (139/141)	23,953	141
Sampling	0.582	22 % (30/139)	14 % (19/139)	–	122,472	139

Hyper-volume indicator—result of the hyper-volume indicator for each optimization process. *Coverage metric*—for each Pareto front approximation in a row, the values specify the number of found solutions that are not dominated by any solution within the result set of Pareto front approximation specified in the column. *Number of simulations*—number of simulation runs within the whole optimization process. *Number of non-dominated solutions*—number of resulting non-dominated solutions

Note that the behavior of the malicious nodes is different to the selective forwarding attack—the malicious nodes delay randomly all packets within the interval $\langle 0, 5\rangle$. The values of *time window* (p_5) varies between 55 and 295 s (131 s on average), the *minimum received votes* (p_6) varies between 1 and 2 (1.23 on average) and the voting threshold varies between 0.17 and 0.99 (0.77 on average). The *delay timeout* (p_8) was set to 1 s in each solution—packets forwarded by benign node are usually transmitted within this timeout in our test case.

7.3 Robustness Evaluation

Various attacker strategies discussed in Sect. 5 are evaluated in this section. Their settings was presented in Sect. 6.4. We label each of the evaluated case "Random 2 %/10 %", "Centre 2 %/10 %" and "Line 2 %/10 %" for the random, the center drop and the direct centre attacker strategy, respectively. The numbers denote the percentage of malicious nodes in the network. All Pareto front approximations were computed using NSGA-II set up according to "Evo #2". We compare all the results using hyper-volume indicator.

Single Pareto Front Approximation Evaluation on Multiple Attacker Strategies Table 4 summarizes all performances of the IDS settings in Pareto front approximation pf_i optimized for an attacker strategy as_i specified in a row in another attacker strategy as_j specified in a column j.

We can see that any IDS settings in the case with 2 % randomly deployed malicious sensor nodes evince the best hyper-volume indicator comparing to the other attacker strategies, while the case with 2 % malicious sensor nodes surrounding the BS evinces the worst hyper-volume indicator. We found out that while the memory

Table 4 Hyper-volume indicator results for various deployment strategies

	R 2 %	R 10 %	C 2 %	C 10 %	L 2 %	L 10 %	Average diff
Random 2 %	**0.583**	0.520	0.453	0.487	0.534	0.462	0.0277
Random 10 %	0.567	**0.555**	0.465	0.510	0.540	0.504	0.0107
Centre 2 %	0.562	0.539	**0.485**	0.507	0.539	0.495	0.0130
Centre 10 %	0.565	0.539	0.473	**0.515**	0.533	0.499	0.0135
Line 2 %	0.565	0.494	0.461	0.473	**0.554**	0.406	0.0420
Line 10 %	0.567	0.545	0.466	0.510	0.537	**0.513**	0.0112
Average	0.568	0.532	0.467	0.500	0.540	0.480	

For each strategy A in a row, the values specify the hyper-volume indicator of the IDS settings optimized for A in another deployment strategy B specified in a column. Looking to the table from the other perspective, for each strategy A in a column, the values specify the hyper-volume indicator of the IDS settings optimized for a strategy B specified in a row but evaluated in the strategy A. The values in bold states for the best results for each of the strategy—IDS settings optimized and evaluated in the same deployment strategy. The last column specifies the average difference of an IDS optimized for an attacker strategy in a row to the best result achieved for each of the strategy in the column

consumption is constant and the number of false positives do not change significantly, more significant changes in the number of false negatives can be observed. This is caused by the placement of malicious nodes—it is more difficult to detect a malicious sensor node surrounded by other malicious sensor nodes or a malicious node close to the edge of the WSN receiving packets from only one descendant.[3]

See Fig. 10 for an example of IDS optimized for Random 2 % in the attacker strategy Random 10 % (a–c) and vice versa (d–f). While the memory consumption is constant and the number of false positives do not change significantly, we can see more significant changes in the number of false negatives due to higher number of "close-to-edge" sensor nodes.

Multiple Pareto Front Approximations Evaluation On Single Attacker Strategy In Table 4, we can see for any attacker strategy as_j in a column j how each hyper-volume indicator of the IDS settings pf_i optimized for each strategy as_i in a row i differs to the hyper-volume indicator of the IDS settings pf_j evaluated (and also optimized) in the as_j. In the last column of the Table 4, we present the average difference to the best value for each Pareto front approximation pf_i across all attacker strategies. The lower the average difference, the more robust is the Pareto front approximation in another attacker strategies.

We can see that Pareto front approximation of IDS settings optimized for attacker strategy "Random 10 %" performs the best across all other strategies followed by

[3]Such traffic can be overheard by less (if any) number of neighbors comparing to a sensor node placed closer to the BS receiving packets from several directions.

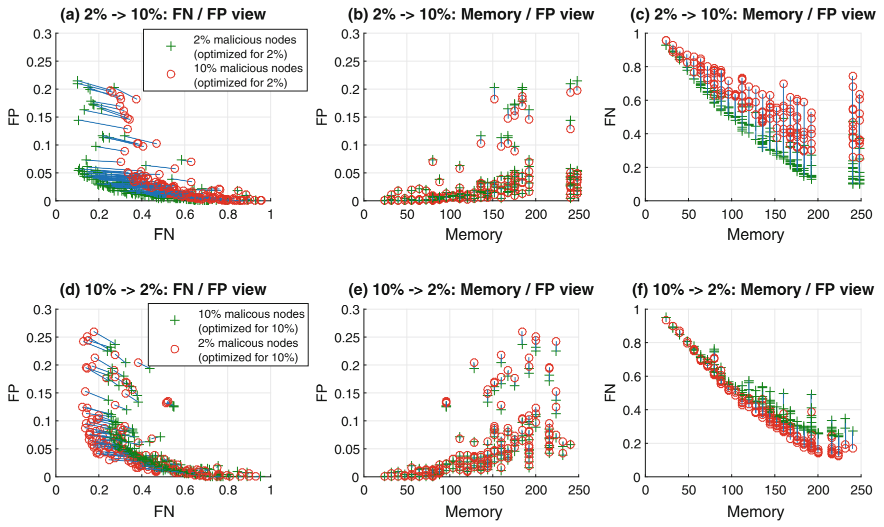


Fig. 10 Influence of changed percentages of malicious sensor nodes on performance of each optimized IDS setting in the random attacker strategy. Green crosses represent IDS performance in an environment for which the IDS was optimized (2 % for Figures (**a**–**c**) and 10 % for Figures (**d**–**f**)). *Red circles* represent IDS performance in an environment with increased [Figures (**a**–**c**) and decreased (Figures (**d**–**f**)] percentage of malicious sensor nodes. *Lines* connect equal IDS settings

"Line 10 %". On the other hand, the solutions found for the attacker strategy "Line 2 %" performs the worst on average in the other strategies.

8 Conclusion

We proposed and implemented a complex optimization framework consisting of an optimization engine and a simulator. The simulations can be executed on multiple computers in a distributed manner. This optimization framework is aimed at but not limited to optimization of intrusion detection systems to detect different types of active attacks on a WSN. In the simulator, the target WSN can be specified in details including the environment, topology, physical properties of the sensor nodes, routing and application.

In this work, we demonstrated usability of the optimization framework on the selective forwarding attack and the delay attack detection. We have shown that efficient Pareto front approximation can be found using multi-objective evolutionary algorithm in a reasonable time. Four different metrics were used to evaluate the optimization processes. The diversity of the non-dominated solutions can provide a network operator with an option to choose any solution according to requirements that can be changed during the WSN operation time.

We discussed thoroughly attacker strategies of the selective forwarding and the delay attack as well as usability of our detection techniques for each variation of any attacker strategy. The IDS was optimized to six different deployments of malicious sensor nodes and the resulting non-dominated IDS solutions were evaluated for robustness on each of the deployment.

The optimization framework can be used directly for, e.g., IDS that we implemented within our security middleware for WSNs—"WSNProtectLayer" [19].

Acknowledgements We would like to thank Ludek Smolik, Lukas Sekanina and colleagues from CRoCS for the discussions and suggestions. This work was supported by the Czech research Project VG20102014031, programme BV II/2—VS. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

References

1. da Silva, A.P.R., Martins, M.H.T., Rocha, B.P.S., Loureiro, A.A.F., Ruiz, L.B., Wong, H.C.: Decentralized intrusion detection in wireless sensor networks. In: Proceedings of the 1st ACM International Workshop on Quality of Service & Security in Wireless and Mobile Networks, pp. 16–23 (2005)
2. Stehlik, M., Matyas, V., Stetsko, A.: Towards better selective forwarding and delay attacks in wireless sensor networks. In: Proceedings of the 13th IEEE International Conference on Networking, Sensing, and Control (2016)
3. Karlof, C., Wagner, D.: Secure routing in wireless sensor networks: attacks and countermeasures. AdHoc Netw. J. **1**(2), 293–315 (2003)
4. Krontiris, I., Dimitriou, T., Freiling, F.C.: Towards intrusion detection in wireless sensor networks. In Proceedings of the 13th European Wireless Conference (2007)
5. Tiwari, M., Arya, K.V., Choudhari, R., Choudhary, K.S.: Designing intrusion detection to detect black hole and selective forwarding attack in WSN based on local information. Proceedings of the 2009 Fourth International Conference on Computer Sciences and Convergence Information Technology. ICCIT '09, pp. 824–828. IEEE Computer Society, Washington, DC (2009)
6. Hai, T.H., Huh, E.: Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge. In: Seventh IEEE International Symposium on Network Computing and Applications, pp. 325–331 (2008)
7. Liu, F., Cheng, X., Chen, D.: Insider attacker detection in wireless sensor networks. In: INFOCOM 2007. 26th IEEE International Conference on Computer Communications, pp. 1937–1945. IEEE (2007)
8. Khanna, R., Liu, H., Chen, H.H.: Self-organization of sensor networks using genetic algorithms. In: IEEE International Conference on Communications, 2006. ICC'06, vol. 8, pp. 3377–3382 (2006)
9. Khanna, R., Liu, H., Chen, H.H.: Dynamic optimization of secure mobile sensor networks: a genetic algorithm. In: IEEE International Conference on Communications, 2007. ICC'07, pp. 3413–3418, (2007)
10. Khanna, R., Liu, H., Chen, H.H.: Reduced complexity intrusion detection in sensor networks using genetic algorithm. In: IEEE International Conference on Communications, 2009. ICC'09, pp. 1–5 (2009)
11. Heady, R., Lugar, G., Servilla, M., Maccabe, A.: The Architecture of a Network Level Intrusion Detection System. Technical report, University of New Mexico, Albuquerque, NM (1990)

12. Stehlik, M., Saleh, A., Stetsko, A., Matyas, V.: Multi-objective optimization of intrusion detection systems for wireless sensor networks. In: Li, P., et al. (eds.) Advances in Artificial Life, ECAL 2013, Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems, pp. 569–576. MIT Press, Cambridge, MA (2013)
13. Banerjee, S., Grosan, C., Abraham, A.: IDEAS: intrusion detection based on emotional ants for sensors. In: Proceedings of 5th International Conference on Intelligent Systems Design and Applications, 2005. ISDA '05, pp. 344–349. IEEE (2005)
14. Banerjee, S., Grosan, C., Abraham, A., Mahanti, P.K.: Intrusion detection on sensor networks using emotional ants. Int. J. Appl. Sci. Comput. **12**(3), 152–173 (2005)
15. Mukherjee, P., Sen, S.: Using learned data patterns to detect malicious nodes in sensor networks. In: Proceedings of the 9th International Conference on Distributed Computing and Networking. ICDCN'08, pp. 339–344. Springer, Berlin (2008)
16. Roosta, T., Shieh, S., Sastry, S.: Taxonomy of security attacks in sensor networks and countermeasures. In: The First IEEE International Conference on System Integration and Reliability Improvements, vol. 25, p. 94 (2006)
17. Loo, C.E., Ng, M.Y., Leckie, C., Palaniswami, M.: Intrusion detection for routing attacks in sensor networks. Int. J. Distrib. Sens. Netw. **2**(4), 313–332 (2006)
18. Stetsko, A., Smolka, T., Matyas, V., Stehlik, M.: Improving intrusion detection systems for wireless sensor networks. In: Boureanu, I., et al. (eds.) Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 8479, pp. 343–360. Springer, Berlin (2014)
19. Matyas, V., Svenda, P., Stetsko, A., Klinec, D., Jurnecka, F., Stehlik, M.: Securing Cyber Physical Systems, chapter 5: WSNProtectLayer Security Middleware for Wireless Sensor Networks. CRC Press, Boca Raton, FL (2015). ISBN 978-1-4987-0098-6
20. Roman, R., Lopez, J., Gritzalis, S.: Situation awareness mechanisms for wireless sensor networks. IEEE Commun. Mag. **46**(4), 102–107 (2008)
21. Anderson, D.P.: BOINC: a system for public-resource computing and storage. In: Proceedings of IEEE/ACM Workshop on Grid Computing, pp. 4–10 (2001)
22. Köpke, A., Swigulski, M., Wessel, K., Willkomm, D., Klein Haneveld, P.T., Parker, T.E.V., Visser, O.W., Lichte, H.S., Valentin, S.: Simulating Wireless and Mobile Networks in OMNeT++ the MiXiM Vision. In: Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, Simutools '08, pp., 71–78, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels (2008)
23. OMNeT++. OMNeT++ Network Simulation Framework—Homepage. http://www.omnetpp.org/. Accessed 22 Oct 2015
24. Stetsko, A., Stehlik, M., Matyas, V.: Calibrating and comparing simulators for wireless sensor networks. In Proceedings of the 8th IEEE International Conference on Mobile Adhoc and Sensor Systems, pp. 733–738. Los Alamitos (2011)
25. Rappaport, T.: Wireless Communications: Principles and Practice, 2nd edn. Prentice Hall PTR, Englewood Cliffs, NJ (2001)
26. Crossbow. TelosB Datasheet. http://www.willow.co.uk/TelosB_Datasheet.pdf. Accessed 26 Oct 2015
27. Talbi, E.G.: Metaheuristics—From Design to Implementation. Wiley, New York (2009)
28. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
29. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report, Eidgenössische Technische Hochschule Zürich (ETH) (2001)
30. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point. In: Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms. FOGA '09, pp. 87–102. ACM. New York, NY (2009)
31. Fonseca, C.M., Paquete, L., Lopez-Ibanez, M.:. An improved dimension-sweep algorithm for the hypervolume indicator. In: IEEE Congress on Evolutionary Computation, 2006. CEC 2006, pp. 1157–1163 (2006)

32. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)
33. Jurnecka, F., Stehlik, M., Matyas, V.:. On node capturing attacker strategies. In: Security Protocols XXII—22nd International Workshop Cambridge. Revised Selected Papers, pp. 300–315. Springer LNCS (2014)
34. Yu, B., Xiao, B.: Detecting selective forwarding attacks in wireless sensor networks. In 20th International Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. IEEE (2006)