A Robot-Sensor Network Security Architecture for Monitoring Applications

Francisco José Fernández-Jiménez, and José Ramiro Martínez-de Dios

***Abstract*—This paper presents SNSR (*Sensor Network Security using Robots*), a novel, open, and flexible architecture that improves security in static sensor networks by benefiting from robot-sensor network cooperation. In SNSR, the robot performs sensor node authentication and radio-based localization (enabling centralized topology computation and route establishment) and directly interacts with nodes to send them configurations or receive status and anomaly reports without intermediaries. SNSR operation is divided into stages set in a feedback iterative structure, which enables repeating the execution of stages to adapt to changes, respond to attacks, or detect and correct errors. By exploiting the robot capabilities, SNSR provides high security levels and adaptability without requiring complex mechanisms. This paper presents SNSR, analyzes its security against common attacks, and experimentally validates its performance.**

***Index Terms*—sensor network, robot, security.**

I. Introduction

A WIRELESS sensor network (WSN) is a set of spatially dispersed devices equipped with sensing, computational, and wireless communication capabilities that can self-organize and collaborate to perform tasks. WSNs are used in many different fields [1] and are deeply dependent on the application. A large diversity of types of WSN arise with different requirements and characteristics including device capabilities (homogeneous or heterogeneous), static or dynamic entities, different topologies (flat or clustered), among many others. This work focuses on WSNs used in the monitoring of industrial settings or civil infrastructures, where sensor nodes gather measurements and collaborate using ad-hoc communications to deliver them to a sink node, also known as gateway or base station (BS). All the sensor nodes have the same role (flat architecture), and once deployed remain static (static WSN) except for occasional location changes, or node addition/deletion.

Security is critical in many monitoring applications. WSN has a large attack surface due to a variety of vulnerabilities [2] originated mainly by: a) nodes restricted energy and computational resources, which constrain their functionalities; b) deployment in hostile environments often at unknown positions and with difficult accessibility, which hinders node repair; c) unattended operation and remote network management that hamper physically checking the nodes' status, and simplify performing physical (or planning complex) attacks; d) high node exposure to disconnection, destruction, or modification; e) easy access to the wireless channel to eavesdrop or inject packets; f) use of an ad-hoc network infrastructure, where nodes trust each other; g) use of a network topology with frequent interchanges of routing information that might be false; h) unreliable communication, which hinders distinguishing between attacks and errors; and e) use of distributed algorithms involving many nodes, which are prone to internal attacks. A wide diversity of types of attacks take advantage of these vulnerabilities, see e.g., [2]–[4]. Attacks can be classified as: active (e.g., jamming) or passive (e.g., eavesdropping); internal (e.g., selective forwarding) or external (e.g., packet alteration); according to the communication layer; and according to the objective, in which they are classified in service availability and integrity (e.g., node destruction, black hole), privacy and secrecy (e.g., traffic analysis, information disclosure), and data integrity (e.g., packet injection, node replication).

Francisco J. Fernández-Jiménez is with the Depto. de Ing. Telemática, Universidad de Sevilla, 41092 Sevilla, Spain (e-mail: fjfj@us.es).

J. Ramiro Martínez-de Dios is at the GRVC Robotics Lab Sevilla, Universidad de Sevilla, 41092 Sevilla, Spain (e-mail: jdedios@us.es).

Manuscript received xxxxxx yy, 2021; revised xxxxxx yy, 2021.

Fig. 1. Pictures of SNSR validation experiments in status monitoring of a bridge (right) and a cement kiln (left), location omitted for confidentiality.

Among the main security requirements that WSN should fulfill are availability, confidentiality, integrity, authenticity and data freshness, node authentication and authorization, forward and backward secrecy, and secure localization of critical nodes [2], [3]. The main security countermeasures can be classified as preventive, intrusion detection systems (IDS), and reactive [4]. Preventive countermeasures strengthen the WSN against certain attacks before they occur. Typical examples are: encryption to protect privacy; digital signatures for authentication; message authentication code (MAC) for data integrity and authentication; timestamps for data freshness; channel surfing and frequency-hopping spread spectrum to avoid interference; location of nodes to avoid intrusion and false information; or randomized communications to avoid traffic analysis. IDS identifies attacks when they occur by using techniques based on inspection of network traffic, observing the RSSI (received signal strength indicator) of the interchanged packets, detecting abnormal behaviors or suspicious patterns and their analysis with different methods, e.g., Dempster-Shafer theory or machine learning. Finally, reactive solutions usually complement an IDS and are activated after the attack to recover from it and prevent further attempts. They usually require changing the configuration of the nodes, revoking/regenerating keys, or expelling malicious nodes from the network. A complete security architecture should include countermeasures of all types. Most security architectures adopt observation-configuration structures that use the anomalies observed by the entities to detect attacks and determine suitable re-configurations to mitigate the attack effects.

Most existing works focus on providing solutions to specific attacks. However, the attack surface is vast, and many mechanisms and protocols are necessary to protect a WSN. Using these mechanisms in the same WSN increases complexity and resource demands, limiting practical feasibility. Many methods adopt centralized approaches in which the entities with more resources, usually the BS, perform the most complex tasks, which also reduces the influence of possible compromised nodes in decision making. However, these approaches increase the dependence on the central node and require routes to reach the BS, whose establishment can also be attacked. Some examples are: work [5], which proposes an intrusion-tolerant tree-structured routing protocol; or [6], which uses a centralized routing protocol to detect wormhole attacks. The Software-Define Network (SDN) paradigm has also been applied in WSN [7], [8], where the control plane, which makes routing decisions, is decoupled from the data plane, which performs packet forwarding. Other examples are centralized IDS, particularly the data analysis and attack detection modules [8], [9], and the use of a trust center for key distribution and generation [10], [11].

A robot is a mobile entity endowed with computational, sensing, and communication capabilities. Recently, aerial robots have become common tools in monitoring tasks such as inspection, failure detection, or predictive maintenance, and this trend will increase in the near future [12]. Robots have obtained remarkable results in helping to solve a wide variety of problems in WSN [13] including node deployment, WSN repairing, recharging of batteries, node localization, or data collection. However, robot-sensor network cooperation for improving WSN security is still under-researched [14].

This paper presents SNSR (*Sensor Network Security using Robots*), a novel, generic, open, and flexible architecture that improves security in static sensor networks by exploiting robot-network synergies. SNSR adopts a centralized iterative approach in which the anomalies observed by all entities are used to detect attacks and determine re-configurations to mitigate their effects, and hence improve network performance. The robot performs sensor node authentication and radio-based localization enabling centralized network discovery and route establishment. It interacts with nodes using short-distance communications, sends them configurations, receives status and anomaly reports without intermediaries, detects anomalies, and can be commanded to obtain direct observations to confirm/discard other anomalies. The integration of the robot within SNSR improves attack prevention, detection, and mitigation without requiring complex or resource-demanding mechanisms. First, as proven in Props. (1)-(3) in Appendix, the use of the robot in SNSR provides enhanced anomaly observation (hence, attack detection) and mitigation. Second, it enables attack prevention mechanisms such as the use of short-range communications, network discovery and topology establishment with authenticated nodes, centralized route calculation without interference, direct collection of node status without intermediaries, among others. In SNSR, these mechanisms substitute others that in traditional architectures are often computed in a distributed manner or use preset information, largely reducing the attack surface and node resource consumption. SNSR constrains node behavior but keeps high flexibility in the mechanisms and protocols, complementing a wide range of existing countermeasures without imposing protocols, optimization criteria, or algorithms. It has been implemented using widely-adopted well-proven secure protocols, and validated in challenging scenarios, see Fig. 1.

The contributions of this paper are:

1) SNSR architecture, whose iterative observation-reconfiguration structure using the robot enhances attack prevention, detection, and mitigation.
2) Protocols and mechanisms for SNSR in instead of others used in traditional architectures that are prone to attacks.
3) Implementation using existing well-proven secure protocols, evaluation, and experimental validation in challenging outdoor scenarios.

This paper is structured as follows. Section II summarizes the main related works. SNSR is presented in Section III. Its operation and main modules are summarized in Sections IV and V. SNSR implementation and experimental evaluation are described in Sections VI and VII. Section VIII compares SNSR with related work. The conclusions are summarized in Section IX. The advantages of SNSR over architectures without robot are formally analyzed in Appendix. Experimental data are provided as additional material [15].

II. Related Work

Security measures include prevention, detection, and mitigation mechanisms. Due to the large attack surface of WSN, most works focus on a particular security aspect. The following summarizes the works more directly related to SNSR. Table I shows the taxonomy of the presented related works classified according to their security topic, defense mechanisms, and the use of centralization and mobile robots.

Cryptography is the base of the main proactive countermeasures used in WSN [2], [3]. There are two main cryptographic systems: symmetric and asymmetric. Symmetric cryptography is efficient but requires complex procedures for key management. Several surveys [35], [36] have pointed out the difficulties of existing key management schemes. Besides the loss of privacy, disclosure of symmetric keys is dangerous since many systems interpret their possession as implicit authentication. Asymmetric cryptography, also called public key cryptography (PKC), simplifies and makes key distribution more secure but is usually more computationally expensive. The most widely adopted algorithms are Rivest–Shamir–Adleman (RSA) and elliptic curve cryptography (ECC). ECC is more efficient than RSA and requires shorter keys to achieve a similar security level. Work [16] implemented ECC in low-resource sensor

TABLE I
TAXONOMY OF RELATED WORK

Reference	Topic	P	D	M	C	R	Highlighted features
[16]	Cryptography	✓	×	×	×	×	Distributed PKI implemented with ECC in low-resource sensor nodes
[17]	Cryptography	✓	×	×	×	×	Hardware implementation of cryptographic algorithms in sensors, including ECC
[18]	Cryptography	✓	×	×	×	×	Revision of WSN protocols using ECC, new protocol that stores prevalidated public keys in nodes
[19]	Cryptography	✓	×	×	×	×	ECC is used for distributed group key management
[20]	Cryptography	✓	×	×	✓	×	ECC signature aggregation, an adaptation of certificateless PKC to WSN.
[21]	Standard	✓	×	×	✓	×	Standard and commercial architecture with protocols using both PKC and symmetric cryptography
[22]	Cryptography	✓	×	×	✓	×	Guidelines to reduce the overheads of PKI in IoT
[23]	Cryptography	✓	×	×	✓	×	Description of the key escrow problem and its solution with a self-certified key distribution scheme
[10]	Cryptography	✓	×	✓	✓	×	Adaptation of identity-based PKC to WSN
[9]	IDS	×	✓	×	✓	×	Centralized anomaly collection, anomaly-based IDS
[24]	IDS	×	✓	×	✓	×	Anomaly detection performed on all nodes
[25]	IDS, Routing	✓	✓	✓	×	×	Anomaly detection and analysis performed only on selected nodes
[26]	IDS	×	✓	×	✓	×	Only BS detects attacks by analyzing network traffic, misuse-based IDS
[27]	IDS	×	✓	×	✓	×	Detection is performed heterogeneously depending on the resources of each entity
[28]	IDS	×	✓	×	×	×	Distributed collection and detection and big data processing
[8]	IDS	×	✓	✓	✓	×	Distributed anomaly collection, attack detection using energy prediction models
[29]	IDS	×	✓	×	✓	×	IDS specific for low-rate denial of service attacks on routing protocols
[30]	IDS	×	✓	×	×	×	IDS based on trust and reputation with environmental parameters
[5]	Routing	✓	×	×	✓	×	Secure routing protocol where the BS calculates all routes and neighbors of all nodes
[6]	Routing	✓	✓	✓	✓	×	Centralized reactive routing protocol for detecting only wormhole attacks
[7]	IDS	×	✓	×	✓	×	Review of specific attacks on SDN in WSN, specialized IDS for SDN
[31]	CH election	✓	×	×	✓	✓	UAV-assisted centralized election of cluster heads (CH) to avoid interference from malicious nodes
[32]	Data collection	×	✓	×	✓	✓	A UAV is employed to collect data and take direct measurements
[33]	Cryptography	✓	×	×	✓	✓	Geometry-based distribution of shared keys to a group of nodes using a mobile robot
[34]	Cryptography	✓	×	✓	✓	✓	The use of a UAV as a key distribution and coordination center for public keys
SNSR	Security architecture	✓	✓	✓	✓	✓	This work

P: Prevention mechanisms D: Detection mechanisms M: Mitigation mechanisms C: Centralized mechanisms R: Use of mobile robots ✓: Yes ×: No

nodes, and [17] developed a hardware implementation achieving drastic time and resource reductions. Work [18] reviewed the methods that secure WSN communications using ECC and proposed an enhanced security protocol for heterogeneous WSN based on ECC authentication. In [19] ECC was used for authenticated key agreement in a protocol for distributed group key management. In [20] messages were signed individually with ECC, and the signatures of different messages were aggregated into a single signature that is easier to verify, reducing communication and computational cost. Many protocols use PKC mostly for initial authentication and session key agreement and then change to symmetric cryptography once the keys have been exchanged. This approach is adopted by *ZigBee Smart Energy* [21], IPSec (for IP), TLS (for TCP), and DTLS (for UDP).

An additional challenge of PKC is how to authenticate the association between an identity and a public key, and how to revoke it if necessary. In Public Key Infrastructure (PKI), the most used approach, certificates are generated with the above pairing and a serial number digitally signed by a Certificate Authority (CA). Certificates must be stored, transmitted, and verified, which consumes resources. Revocation involves distributing lists of revoked certificate serial numbers or querying the CA or another trusted authority. Work [22] discussed the use of certificates in DTLS, and provided guidelines to reduce its computational cost e.g., by using pre-verification or session resumption. In identity-based cryptography, the public key is the identity itself, so it is not necessary to transmit it. However, it requires a trusted third-party Private Key Generator (PKG) that generates all the private keys and this leads to a private key escrow problem that is addressed in [23]. It has a higher computational cost than PKI due to the pairing process (bilinear association of an identity to a point on the ECC curve), and revocation is more complex as it cannot reuse identifiers. Certificateless cryptography is a variant of the previous one, which solves the key escrow problem but again requires the exchange of public keys as they are not directly derivable from the identity, and still incurs in high computational costs. Recently, works [10], [20] presented adaptations to WSN of the above alternatives to PKI by avoiding the pairing process but limiting its functionality to the digital signature operation.

Many security systems are based on the detection of attacks and the identification and location of the attacker. This is the case of Intrusion Detection Systems (IDS), which operation is usually divided into information collection, detection model, and response decision [9]. Network monitoring and anomaly detection can be done on all nodes [24], on selected nodes [25], centralized by analyzing network traffic [26], or heterogeneously depending on the resources of each entity [27], among others. Anomaly collection can be centralized at one entity [9] or distributed [8], [28]. In the latter, nodes only have local knowledge of the network, often resulting in lower attack detection rates. There are detectors for different types of attacks or specific attacks such as denial of service [29] or clone nodes [37]. Depending on the type of information analyzed, IDSs can be divided into [38]: a) anomaly-based [9]; b) misuse-based [26]; and c) specification-based, which detect attacks using manually preset rules and specifications, and have the advantages of both previous categories but developing the specifications can be costly in complex systems. Many methods have been proposed to infer attacks using pattern matching, Machine Learning (e.g., Support Vector Machine or neural networks), energy prediction models [8], big data [28], trust and reputation [30], among others. Some conclusions from existing IDSs can be highlighted. First, anomaly detection and transmission can require high energy consumption for sensor nodes. Anomaly detection is not perfect, and the effects of false positives/negatives must be considered. Most works focus on the processing of anomalies, but the ability to observe anomalies, and the transmission of anomalies and corrective actions must also be considered. Attackers can generate false anomalies, interfere with anomaly transmission, and even compromise the members of a distributed IDS.

Existing WSN security works present distributed or centralized solutions. Load balancing and sharing and the absence of a single point of failure (SPOF) are advantages of distributed schemes. However, they are based on local information, have higher computational costs on nodes with limited resources, higher message exchange, and in general, the probability of a malicious node influencing the system is higher. Centralized schemes usually perform the most demanding tasks at the entities with more resources or higher reliability (typically BS) but present a SPOF and a bottleneck. In the basic version of INtrusion-tolerant routing protocol for wireless SEnsor NetworkS (INSENS) [5], BS calculates the routes from all nodes to BS and the neighbors of all nodes using the local topological information provided by nodes. In work [6], the routes are calculated for detecting wormhole attacks. The SDN paradigm has been applied to WSN by allowing nodes to receive commands on how to process packets from a controller, see [39]. SDN makes network management more flexible and simpler, but nodes must first discover routes to the controller by other means, and the communication with the controller must not be interrupted. However, new attacks arise from these requirements, see e.g., [7]. Table I identifies other works with some form of centralization such as [20], [31] and centralized IDSs such as [7], [9], [26].

Mobile robots have been used to help WSN in a wide variety of problems, e.g., coverage extension, relaying, data aggregation, distribution and collection, and node deployment, replacement, or localization [13], [14], [40]. In many works that integrate mobile robots in WSNs, security is a secondary objective or a side effect. Work [31] used an Unmanned Aerial Vehicle (UAV) to select the CHs of a WSN after collecting the nodes energy status reducing the probability of a malicious node being elected as a CH, but it did not consider other attacks, e.g., during the initial key establishment phase. Work [32] proposed a UAV for data collection. The UAV takes direct measurements to detect node misbehavior. The use of robots to improve WSN general security has focused on key distribution, without using the robot in other security tasks. In work [33], a mobile element was used to distribute shared keys to a group of nodes, seeking to minimize the dissemination zones and the number of keys without connectivity loss. Work [34] used a UAV as a key distribution and coordination center for public keys. Nodes request peers' public keys from the UAV.

This paper proposes a new security architecture that exploits tight sensor network-robot cooperation to achieve high security levels with simple and efficient security mechanisms. Robot-WSN cooperation endows the SNSR with enhanced attack detection and mitigation capabilities (see Appendix), and also strengthens prevention by replacing mechanisms that in traditional architectures are prone to attacks.

III. SNSR Security Architecture

Assume a number of static sensor nodes deployed in a monitoring application. Nodes gather, filter, and transmit measurements to a base station (BS) –also static, where they are logged, processed, and/or transmitted to a remote center. Nodes are often installed at locations with difficult accessibility. We assume that an aerial robot (R) is used as part of the monitoring plan, performing inspection, failure detection, or predictive maintenance. SNSR requires the participation of the robot only at occasional times; during the rest of the time, it can be used in other tasks. The robot communicates directly with the nodes using the sensor network protocol stack. It can also communicate with BS just as with any other node or using a longer-range higher-bandwidth network. The network, communication channel, and scenario are assumed realistic. Nodes are endowed with low computational and transmission capabilities. The network is not very dynamic: nodes have low failure probability and new nodes are not frequently added to the network. Packet loss –caused by collisions, interference, or noise, among others– is not negligible and varies along the scenario. The link layer (LL) at each entity checks that the received message is correct, or discards it otherwise. Nodes operate in unattended manner and are exposed to malicious attacks. The BS is at a secure location and cannot be easily tampered or cloned. The robot can only be tampered if captured during the flight, which can be quickly detected. Also, Denial of Service (DoS) or jamming attacks cannot continue constantly without being detected and eliminated.

SNSR is a generic and flexible architecture that improves security by exploiting network-robot cooperation. It is not particularized for any concrete network stack and can be easily adapted to existing networks.

It is devised as an extension that defines, complements, and modifies the services in the management and security planes of the protocol stack, while providing high flexibility and adaptability in the communication plane –including protocols, security mechanisms, cryptographic algorithms, and message coding in each layer. The robot (R) provides interesting advantages in SNSR:

- R discovers, localizes, and authenticates the nodes. Nodes can only establish communication with entities that have been previously authenticated by R. Hence, R secures the initial configuration stages, critical in the network performance and prone to security attacks.
- Network topology and route establishment are performed centrally using topological information from authenticated nodes observed directly by R, avoiding distributed protocols and intermediate nodes, preventing attacks or mitigating its effects.
- R sends directly to each node its configuration –including routes and lists of authenticated neighbors and revoked certificates– avoiding their alteration or discard. A node cannot establish communication to any other authenticated entity except R until it has been configured.
- During the flight, R executes anomaly detection techniques and also collects without intermediaries the anomalies detected by the nodes.

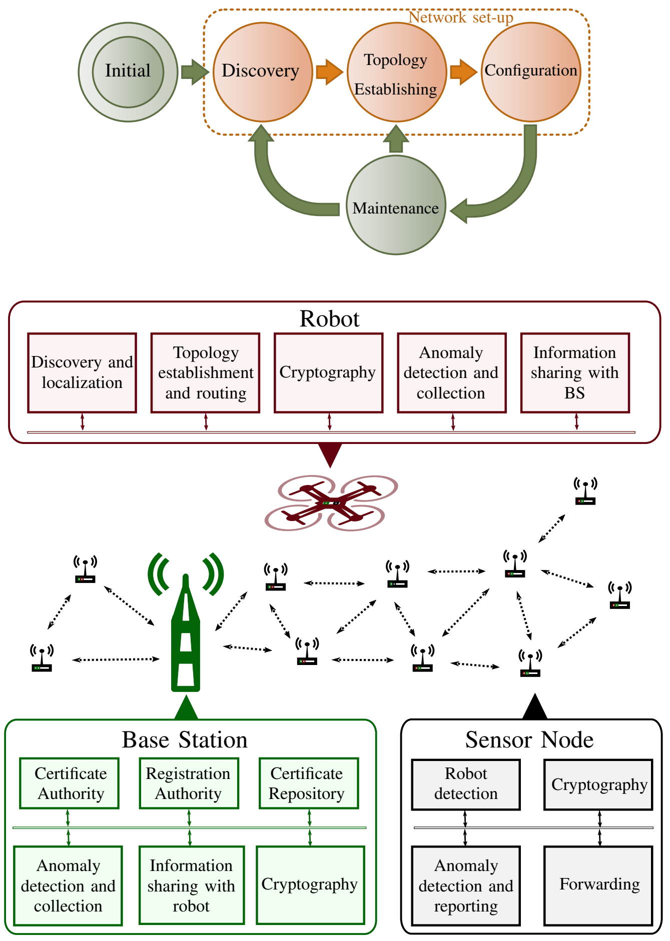

Fig. 2. Top) Main stages in SNSR operation. Bottom) Scheme of SNSR with the main security modules in each entity.

Figure 2-top summarizes the operation of SNSR. In *Discovery*, R flies a safe trajectory to discover and authenticate sensor nodes, and compute their 3D locations. Next, using the collected data, the network topology and routes are determined in *Topology Establishing*. In *Configuration*, R safely flies near each node and sends its configuration directly to it. In *Maintenance*, every entity executes anomaly detection mechanisms and reports the detected anomalies to BS directly or using R. BS analyzes the anomalies and if necessary decides corrective actions, e.g. the execution of a new *Discovery* stage. The security strengths of SNSR mainly consists in: 1) the SNSR iterative structure using the robot, which naturally provides better anomaly observation and anomaly/configuration transportation than traditional centralized WSN without robot (see Appendix); and 2) a set of simple well-tested security mechanisms that exploit robot-WSN operation and substitute methods that in traditional networks are performed in a distributed manner or using preset information. Figure 2-bottom shows the main security modules executed in each entity. Other functionalities necessary for SNSR operation, e.g. robot navigation, are performed. Not being the object of this paper, they are omitted for clarity. SNSR adopts authentication based on PKI. For simplicity, PKI modules –Certificate Authority (CA), Registration Authority (RA), and Certificate Repository (CR) that includes the certificate revocation list (CRL) with the certificates that have ceased to be valid– are executed in BS. They could have also been placed in a remote external element providing stronger security. Each entity executes a cryptography module that includes its CA-issued certificate, its private key, and the CA certificate. SNSR can independently use secure protocols in the LL and/or the transport layer (TL), see Table II, providing capability of adapting to the available node resources and the required authentication, confidentiality, and data integrity levels. The mode is preconfigured in each entity. Mode `SEC_LT` provides security in LL and TL, but requires higher resources. Mode `SEC_LTE` avoids unnecessary double-encryption in LL and TL in end-to-end communications between neighbors. Even in mode `SEC_NO` –without secure LL and TL protocols– the role of R in network set-up provides some intrinsic security, but it can be insufficient in many cases. In `SEC_NO` and `SEC_TE`, R does not authenticate nodes. Although resource efficient, their use is discouraged.

TABLE II
SECURITY MODES IN SNSR

		TL security		
		WITHOUT	EXCEPT NEIGHBORS	WITH
LL security	WITHOUT	`SEC_NO`	`SEC_TE`	`SEC_T`
	WITH	`SEC_L`	`SEC_LTE`	`SEC_LT`

IV. OPERATION

The network topology database (NTDB) stores the updated information of the network: a) initial constraints if any, e.g. the initial CRL or a whitelist with allowed nodes; b) data collected during *Discovery*; and c) configuration resulting in *Topology Establishing*. Its UML Entity-Relationship diagram is shown in Figure 3. It includes: node configuration `NodeData`; BS configuration `BSData`; routing information `RouteData`; parameters required by each node for communicating with a neighbor `NeighborData`; and data to define the CRL `CRLData` and `SerialData`. NTDB is managed by BS and exchanged with R. The configuration resulting in each *Topology Establishing* is stored in a different NTDB `version`.

In SNSR, communication is connection-oriented. Below are the main interactions between already connected entities:

- R⟺BS. Purposes: 1) NTDB exchange; 2) R report of collected anomalies; 3) BS informing R about changes in the CRL; 4) R discovering the BS and asking its status; and 5) BS giving management commands to R.
- R⟺ node N_i. Purposes: 1) R discovering, configuring, and asking status of N_i; 2) N_i reporting anomalies to R.
- BS⟺N_i. Purposes: 1) N_i transmitting the gathered measurements to BS; 2) BS requesting to change N_i configuration and CRL; and 3) N_i reporting anomalies.
- N_i⟺N_j. Nodes only communicate with each other to collaboratively forward packets to the destination.

Table III shows the management application messages between entities. All messages except `ANOM_STATUS` and `SET_CRL_BCAST` employ a reliable transmission service – using loss detection and retransmission. `SET_CRL_BCAST` is the only one broadcast by flooding in NL.

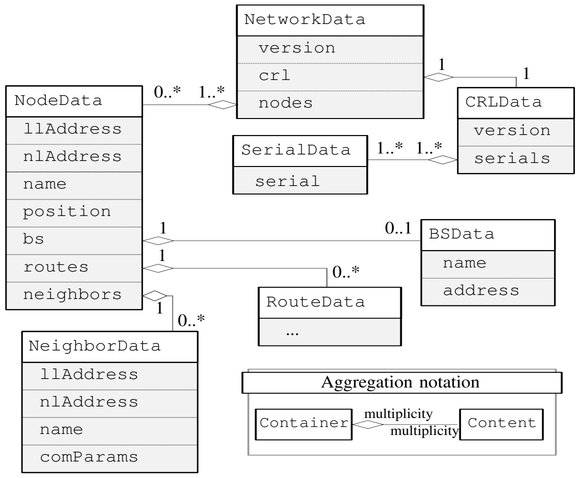

Fig. 3. UML Entity-Relationship diagram of NTDB.

TABLE III
DESCRIPTION OF THE MAIN MANAGEMENT APPLICATION MESSAGES

Request/Response	**Involved entities** (initiator$\Longleftrightarrow$responder) / **Purpose**
STATUS_REQ STATUS_RES	R$\Longleftrightarrow$any, BS$\Longleftrightarrow$N$_i$ / To obtain the entity status
CFG_REQ CFG_RES	BS$\Longleftrightarrow$N$_i$, R$\Longleftrightarrow$N$_i$ / To change configuration or CRL
ANOM_STATUS ANOM_ACK	R$\Longleftrightarrow$BS, N$_i$$\Longleftrightarrow$R,BS / To notify detected anomalies
GET_ND_REQ GET_ND_RES	R$\Longleftrightarrow$BS / To obtain NTDB from BS
SET_ND_REQ SET_ND_RES	R$\Longleftrightarrow$BS / To send NTDB to BS
SET_CRL_BCAST	BS$\Longleftrightarrow$all (by flooding) / To urgently update CRL
MGMT_CMD_REQ MGMT_CMD_RES	BS$\Longleftrightarrow$R / To execute management commands (implementation-dependant).

A. Initial

Each entity is initially configured with its static (e.g. type of entity and identifier) and security settings (e.g. security mode, cipher suites allowed, private key, CA and own certificates and initial CRL –empty by default). BS, assumed secure, contains the initial NTDB, which is empty except for the initial constraints: the initial CRL CRLData and the data of the allowed entities NodeData. Nodes are initially in *Node_Initial* mode: they do not transmit until they detect the presence of a robot. BS also has the bounding coordinates of the scenario –node locations are unknown. The robot take-off/landing location is assumed within the BS communication range. R connects with the BS (as any other connection in SNSR, they use beacons, authenticate one another, and establish a secure channel) and requests the updated and the initial NTDB and the status of BS, see Sections V-C and V-D.

B. Discovery

This stage can be performed after *Initial* or as a decision taken in *Maintenance*. It is executed similarly in both cases. For brevity, only the first is described. BS commands the start of *Discovery* in a MGMT_CMD_REQ message, which includes the scenario bounding coordinates. R computes an obstacle-free trajectory –e.g. a zig-zag pattern– that ensures that all nodes will receive beacons. R takes off and performs the trajectory transmitting beacons to announce its presence. Beacon processing is described in Section V-A. If a node validates the received beacon, it connects to R: they authenticate one another and establish a secure channel, see Section V-B. After connection, R sends a status request message (STATUS_REQ), see Section V-D. Once authenticated, R estimates the 3D location of each discovered entity using a radio-based technique. All collected data is registered in a temporary NTDB. After completing the trajectory, if the whitelist was not provided in *Initial*, R lands. Otherwise, R checks if it collected data from all allowed nodes, and if not, the flight is repeated up to a maximum number of times (e.g. 3). The temporary NTDB will be used to determine the network topology and routes, which can be computed by BS or R. In the former case, R transmits the NTDB to BS using a SET_ND_REQ message.

C. Topology Establishing

In this stage the configuration of each entity is computed to create a fully connected network, which is stored in a new version of the NTDB. It is typically performed at BS, which often has higher computational resources. For small networks, computing onboard R enables saving transmissions and starting the *Configuration* flight without landing, just after the *Discovery* flight. In both cases the steps are the same: 1) node filtering using the whitelist and the CRL; 2) generation of the network topological graph; 3) computation of routes; and 4) assignation of network addresses and other communication parameters. Finally, a new NTDB is built and shared between R and BS using messages SET_ND_REQ or GET_ND_REQ.

Node filtering detects and discards, leaving out of the topology, invalid and cloned nodes. Next, the topological graph G=(V, E), composed of a set of vertexes V (entities) and a set of edges E (each representing adjacency between two entities) is created. If *Topology Establishing* is executed after *Initial*, two nodes are considered neighbors if their distance (using the locations computed in *Discovery*) is lower than a predefined range r. This simple but efficient criterion may lead to topology errors, which will be detected in *Maintenance* as anomalies. If *Topology Establishing* is executed after *Maintenance*, the connectivity information collected from nodes, see Section IV-E, is used. Only nodes reachable from BS are considered: the rest are left unconfigured, and reported to the administrator. To limit resource and energy consumption, each vertex v will be able to communicate with up to N_v authorized neighbors selected with criteria based on link quality, available resources, or type of network (mesh or tree).

SNSR gives flexibility for adopting different routing mechanisms. One option is to compute routes centrally each time this stage is executed, and nodes operate using static routing. This approach has the advantages of static routing (strong security and low bandwidth and computational overhead) and of dynamic routing (easy to be configured and effective at selecting the best route). Besides, it decouples the control plane from the data plane, as in Software Defined Networks (SDN) [41], and different criteria can be used to compute the routes. We adopt this approach in the implemented SNSR using a simple minimum hop route calculation method. Besides, SNSR also supports dynamic routing including: 1) existing protocols without modification –in these cases SNSR provides the additional advantage that all nodes are authenticated; and 2) modifications of existing routing protocols where SNSR adds initial centralized route establishing without intermediaries. SDN-based protocols can be also used. The SDN controller functions are performed by BS or by R and the routes to the controller can be computed in a distributed manner as in [42], [43] or centrally at BS or R. SNSR is not particularized for any network stack. The network layer adopted determines the address allocation method (e.g. flat, hierarchical, topology-based) and other parameters.

D. Configuration

Each node is provided with its configuration (`NodeData`) and CRL (`CRLData`) obtained in *Topology Establishing*. When a node receives its first configuration, its mode changes to *Node_Config*. Nodes are configured by R, which flies next to each node and transmits the node configuration directly. BS could also remotely re-configure nodes that are already in *Node_Config* mode, but this option is not recommended since the messages could be interfered.

BS commands the start of *Configuration* using a `MGMT_CMD_REQ` message. R plans an obstacle-free trajectory such that it can directly communicate with each node, see e.g., [44]. Next, R performs the trajectory transmitting beacons to announce its presence. Beacon processing and connection with R are described in Sections V-A and V-B. R can configure many nodes simultaneously in few ms, see Section V-E. After connection establishment, R requests the node status. If R configuration or CRL versions are different from node versions, it sends a `CFG_REQ` message with its new configuration. When the node receives it, applies the new configuration, eliminates no longer authorized connections, and confirms configuration update sending a `CFG_RES` message. R retries if it does not receive the `CFG_RES` message before a timeout. After configuring all nodes, R lands. During network configuration there is often a transitory period of instability depending on the order in which nodes are configured, but communication with R is direct and unaffected.

E. Maintenance

After *Configuration* the monitoring application starts. Every entity detects anomalies (using methods for detecting connection errors and mechanisms for detecting specific anomalies) and reports them –directly or using R– to BS, which decides the corrective actions. After a node transmits its *CFG_RES* message in *Configuration*, it connects to BS and all its authorized neighbors, and starts detecting anomalies. In every flight, R performs anomaly detection mechanisms using all the information it gathers. Besides, in *Maintenance* it performs additional periodic and on-demand flights in which it localizes and connects to nodes, collects their detected anomalies, and forwards them to BS. R and BS can also ask the status of nodes to find the cause of the anomalies or to check status.

BS is always detecting and collecting anomalies. In *Maintenance* it analyzes them and decides corrective actions as in IDSs, but with interesting improvements. First, R detects anomalies (as well as nodes and BS) and secures the collection of the detected anomalies, providing enhanced anomaly observation as proven in Prop. (1). Besides, BS can on-demand command robot flights to obtain direct observations –and transmit them without intermediaries– to confirm/discard threats. IDSs require complex techniques to infer the type of threat [45]. Although SNSR supports IDSs, its enhanced observation capability enables using simpler approaches. An efficient anomaly management module based on rules that assign corrective actions to anomalies was sufficient to obtain suitable results, as shown in Section VII. Table IV summarizes the main management rules adopted. Threats that can be solved using cryptography and communication errors detected during the configuration transitory period are not considered. The anomalies shown at the top of Table IV trigger the execution of *Discovery* to confirm/discard potential threats, or to acquire new information. Others, shown at the bottom, require actions from the operator, e.g. in case of detecting duplicated identities. The operator can also force administrative actions such as performing a new *Discovery* stage (e.g. after adding new nodes), or a new *Topology Establishing* stage (e.g. after removing nodes). Besides, urgent CRL update, see Section V-H, is used to rapidly forbid communication with an entity, e.g. in case of robot theft since its keys and privileges can be used to modify other entities' configuration. The management rules in Table IV can be easily extended to consider other anomalies and detectors. R delivers node configurations, improving security, see Prop. (2), which combined with the enhanced anomaly observation, see Prop. (1), provides SNSR with enhanced attack recovery, see Prop. (3).

TABLE IV
MANAGEMENT RULES ADOPTED

Observed anomalies that trigger a new *Discovery* stage	**Potential causes and threats**
Node displacement detected by R	Node position change
Node does not connect to BS but connects to its neighbors	Selective forwarding
Disconnection of a node but not of those who depend on it	Selective forwarding, packet injection
Disconnection of a set of nodes with a common intermediate	Intermediate node failure, selective forwarding
Unreachable neighbor, but it is connected with others	Radio-based range estimation error, node displacement
Peer disconnection	Node failure, power loss, tampering, jamming, node displacement
Connection inactivity timer expired	Idem, selective forwarding, network congestion
Continuous negotiation or protocol error during handshake	Misconfiguration, packet injection or alteration, power attack, man-in-the-middle
Increased error and loss rate, frequent re-connections	Jamming, selective forwarding, network congestion, power attack, packet injection or alteration, node displacement
Observed anomalies that require administrative actions	**Potential causes and threats**
Duplicated identity	Misconfiguration, node replication
Bogus beacons	Sleep deprivation torture, power attacks
Bogus broadcast message	Sleep deprivation torture, power attacks, NL DoS, packet injection or alteration
Invalid or revoked certificate	Tampering, node replication
Temporary interruption of the operation of a sensor network	Tampering, power loss, jamming
Management protocol error	Tampering, spoofing, malfunction

V. MODULES

The structure of the main used messages is presented below (|| denotes concatenation). Application layer messages start with an application header (AH) that includes the type of message and a transaction identifier.

$$Beacon : P \parallel ID \parallel SN \parallel NA \parallel CP \tag{1}$$
$$GET\_ND\_REQ : AH \parallel INITIAL \tag{2}$$
$$GET\_ND\_RES : AH \parallel NTDB \tag{3}$$
$$SET\_ND\_REQ : AH \parallel NTDB \tag{4}$$
$$SET\_ND\_RES : AH \parallel ACCEPTED \tag{5}$$
$$STATUS\_REQ : AH \tag{6}$$
$$STATUS\_RES : AH \parallel CFGV \parallel CRLV \parallel NA \parallel NBL \tag{7}$$
$$CFG\_REQ : AH \parallel T \parallel CRL \parallel CFGV \parallel NA \parallel BS \parallel NBL \parallel NO\_NBL \parallel RD \parallel NO\_RD \tag{8}$$
$$CFG\_RES : AH \parallel CFGV \parallel CRLV \tag{9}$$
$$ANOMALY : OBS \parallel ID \parallel TYPE \parallel OBJ \parallel INFO \parallel TS \parallel CFGV \parallel REPT \tag{10}$$
$$ANOM\_STATUS : AH \parallel ANOMALY\_LIST \tag{11}$$
$$ANOM\_ACK : AH \parallel OBS \parallel ID \tag{12}$$
$$SET\_CRL\_BCAST : AH \parallel CRL \parallel SIGN \tag{13}$$

A. Start of Communication with the Robot

The connection with R is a sensitive step. First, R is the only entity that nodes can connect to without prior information. Second, R sends the configuration to each node, determining its subsequent operation. The connection to R is as follows. R transmits beacon frames, see (1), at partially random intervals to prevent accidental or intentional interference. Beacons are the only messages that are broadcast in LL. After the preamble P, it conveys: R identity ID, its certificate serial number SN, its network address NA, and communication parameters CP. Beacons are not encrypted or authenticated: their content is not critical and should be efficiently processed; and R will be authenticated during the connection establishment.

If the receiver of a beacon is already connected to the robot, it discards the beacon. Otherwise, it processes it verifying that: 1) the ID is valid; 2) the SN is not revoked (using the CRL); and 3) the CP is acceptable. Beacon detection mechanisms based on sequential connection establishment attempts can suffer power attacks, e.g. continuous handshakes originating LL exhaustion or denial-of-sleep. To prevent them, we adopt a method based on reservoir sampling algorithm [46] that analyzes the received beacons and maintains a candidate to attempt connection. If a connection attempt fails, another connection to the next candidate is started immediately. Also, minimum and maximum execution times are set for handshakes: the first one to prevent power attacks and, the second, to limit the time trying to connect to a potentially fake robot.

B. Connections between Entities

Communication between entities in LL or TL is connection-oriented. SNSR supports handshakes between neighbors –at LL– and remote –at TL– entities. Two types of handshakes are adopted: basic (without security) and secure. Every entity except R knows the type and identity of the other end when the connection is initiated, since this information is in the entity dynamic configuration. In the secure handshake both ends: 1) negotiate the algorithms for key exchange, authentication, encryption, and MAC generation; 2) authenticate one another; and 3) exchange a random master key using procedures such as Diffie-Hellman [47] or Menezes–Qu–Vanstone [48]. The identity and type of each entity are stored in its digital certificate. Authentication is performed by exchanging and verifying the certificates signed by the CA, and by signing the messages exchanged during the handshake (which also provides integrity). PKI authentication uses asymmetric cryptography –more secure but more computational demanding. Symmetric cryptography is used for encryption and MAC generation during the rest of the communication using keys derived from the master key. Keys should have perfect forward secrecy (PFS), i.e. keys should not be related to previous keys, avoiding pre-shared keys.

The cost of secure handshakes can be high for low-resource nodes. Connections are usually permanent, except those with R, and its cost is amortized over time. As proposed in [22], to address this cost we use: 1) keepalive messages at configured intervals when they have no data to transmit; 2) resumption handshakes for temporary connection loss, which reuse data from the previous connection; and 3) pre-validation of node certificates by R. Additionally, after a handshake, data is transferred with a sequence number for freshness. Depending on the negotiated algorithms, in secure mode, a key set identifier, a nonce, a MAC, and padding are also transferred, and all or some of this data is ciphered, protecting against eavesdropping and packet injection/duplication/alteration.

SNSR supports security protocols that satisfy these features. There are different standards with recommendations on protocols, schemes, components, and primitives. Security protocols as TLS, DTLS, or IKE version 2, use the IEEE 1363-2000 standard and its revisions and extensions whereas, *ZigBee Smart Energy* [21] is based on *Standards for Efficient Cryptography 1* (SEC 1) [49]. SNSR adopts the same mathematically verified, flexible, and upgradeable protocol at both LL and TL.

C. NTDB Exchange

NTDB storage is centralized by BS, and exchanged with R. With message `GET_ND_REQ`, see (2), and using the parameter `INITIAL` (boolean), R can retrieve from BS the initial or current version of the NTDB. Then, BS responds with message `GET_ND_RES`, see (3). R can also propose a new version of the NTDB to BS using message `SET_ND_REQ`, see (4). BS can accept or reject the new NTDB version using parameter `ACCEPTED` (boolean) in message `SET_ND_RES`, see (5).

D. Status Request

Status requests enable verifying the configuration of an entity and its connections, which help to identify the cause of anomalies. R sends a status request –`STATUS_REQ` see (6)– to every entity it connects to. The entity responds with a `STATUS_RES` message, see (7), which includes its: $CFGV$, configuration version; $CRLV$, CRL version; NA, network address; and NBL, a list of `NeighborData` elements of the authorized neighbors the entity currently communicates with. `STATUS_RES` can be extended with other data of interest for the problem, e.g. node battery level or link quality metrics.

E. Configuration Request

This request changes the dynamic configuration of a node, using the content of NTDB as new configuration. Configuration requests can modify: 1) only the CRL (`CRLData` in

NTDB); 2) only the configuration version (`NetworkData`) and node configuration (extracted from `NodeData`); and 3) both of them. Also, the node configuration can be sent in full or in differential modes –i.e. sending only the changes to reduce data transfer. In total there are 5 types of `CFG_REQ` messages, see (8), which include one mandatory field T, the type of message, and the rest are optional: CRL, to add to the existing CRL; $CFGV$, configuration version; NA, network address; BS, BS data; NBL, list of authorized neighbors and its communication parameters; and RD, routing data. In differential configuration NBL is the list of the new authorized neighbors and communication parameters, while NO_NBL is the list of the no longer authorized neighbors and parameters. The same applies for RD and NO_RD. After receiving `CFG_REQ`, the node performs the changes and sends back `CFG_RES`, see (9), with its new $CRLV$ and $CFGV$.

F. Anomaly Detection

SNSR imposes a well-defined operation in which each entity knows what the others can do. Any deviation that can impact security and proper operation is considered an anomaly. Two categories of anomalies have been identified. The first anomalies refer to communications (connection, protocol, and broadcasting errors) and are detected by all entities using the same mechanisms. The second anomalies refer to observations of R, such as duplicated identities or changes in node positions. Table V shows the main types of anomalies considered. Once detected, they are sent to BS for analysis with the management rules shown in Table IV. The SNSR is flexible to incorporate other anomaly detection modules, e.g., for Sybil attacks or jamming, among others.

Each entity temporarily stores the following data from the anomalies it has detected, see (10): OBS, the observer; ID, a sequential unique identifier of the anomaly for the observer; $TYPE$, the type of anomaly; OBJ, the entity causing the anomaly; $INFO$, type-specific information; TS, a timestamp; $CFGV$, the observer configuration version during detection; and $REPT$, the times the anomaly has been observed. A maximum number of anomalies are stored to prevent overflow attacks. Nodes have limited storage capacity. If exceeded, age or priority criteria are used to select which are stored.

TABLE V
MAIN GENERIC TYPES OF ANOMALIES CONSIDERED IN SNSR

Type of anomaly	Description
`CONN_ERROR_TO_INIT`	Expired maximum wait for the handshake without receiving data from peer
`CONN_ERROR_TO_HS`	Expired maximum wait for the handshake after receiving data from peer
`CONN_ERROR_CERT`	Revoked or incorrect certificate
`CONN_ERROR_HS`	Other errors during handshake
`CONN_ERROR_TO_KEEP`	Exceeded maximum time without receiving data or keepalive messages
`CONN_ERROR_PEER`	Peer closes connection
`CONN_ERROR_SHUTDOWN`	Management plane closes connection
`NETWORK_E_BROADCAST`	Validation error in broadcast message
`MGMT_PROTOCOL_ERROR`	Management protocol error
`NAME_ERROR_DUP`	Duplicate identity
`NODE_DISPLACEMENT`	Node displacement

G. Anomaly Reporting

The detected anomalies should be transmitted to BS with low impact on network performance. R works as a store-and-forward router: it receives anomalies from nodes and transmits them –as well as its own anomalies– as soon as it is connected to BS. Nodes report their temporarily stored anomalies to BS or to R at predefined time intervals –or after another interaction– minimizing the number of messages and using minimum-hop paths, see an example in Figure 4. Anomaly notification messages, see (11), include a list with as many as possible anomalies sorted chronologically, see (10). Reliable transport and fragmentation are avoided. The receiver sends an ACK message, see (12), with the identifier of the last anomaly received. After receiving the ACK, the sender removes the acknowledged anomalies from its temporary anomaly list.

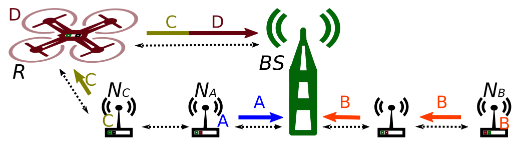

Fig. 4. Anomaly report example: node N_A sends anomaly A directly to BS; B is sent to BS using intermediate nodes (since N_B has no direct connection with BS or R); and N_C sends C to R, which sends C and also D to BS.

H. Urgent CRL Update

BS can send CRL updates to each node (one by one) using `CFG_REQ` messages. However, in critical cases (e.g. robot theft), a mechanism in which BS can fastly send CRL updates to all entities is interesting. In urgent CRL update, BS triggers the flooding of a `SET_CRL_BCAST` message in NL. Each entity, after authenticating the message, forwards it to its neighbors except the sender –more than one transmission in LL is often used to reduce loss probability, e.g. in [50]. Then, BS requests the status of each node and sends `CFG_REQ` messages with the new CRL to non-updated nodes, if any. Urgent CRL updates require secure TL communications with BS. Hence, they are not available in security modes `SEC_NO` and `SEC_L`. Although they consume high bandwidth and use asymmetric cryptography, they are a suitable alternative to one-by-one sending `CFG_REQ` messages for fastly reacting to critical threats that occur occasionally.

The message employed, see (13), includes: CRL, list of the sequence numbers of the certificates to be revoked (in most cases the list includes only one); and $SIGN$, a digital signature that secures the rest of the message and the source address. Field AH includes a unique identifier of the flooding transaction. Entities save the last valid identifier received to prevent forwarding duplicated or old messages. Also, they verify that the sender address is the BS address, and that the digital signature has been made with the public key they obtained in the last connection with BS. Thus, a compromised node sending false messages will be easily detected.

Figure 5 shows a UML sequence diagram that summarizes the operation of SNSR in a simple scenario with 2 nodes (N1 and N2). For clarity, the beacons and lower-level messages are not shown, and the connections with R are established only once and remain uninterrupted along its trajectory. All connection establishments in LL and TL are depicted except the first robot-BS connection, which was performed in *Initial*. The processes that are executed in parallel are shown within parallel operators (`par`). Initially, R is in position ① and is commanded to start *Discovery*. It performs the planned trajectory transmitting beacons, computing node locations, and collecting their status until it reaches position ②. Then, R sends the updated *NTDB* to BS, which performs *Topology Establishing* and creates the new version of the *NTDB*. R asks for the new *NTDB* and BS responses when *NTDB* is ready. Next, R is commanded to start *Configuration*. R starts transmitting the configuration to the nodes, which use it to establish authorized connections. Finally, in *Maintenance* node N1 has a fatal error. N2 detects it as an anomaly and reports it to R, which forwards it to BS.

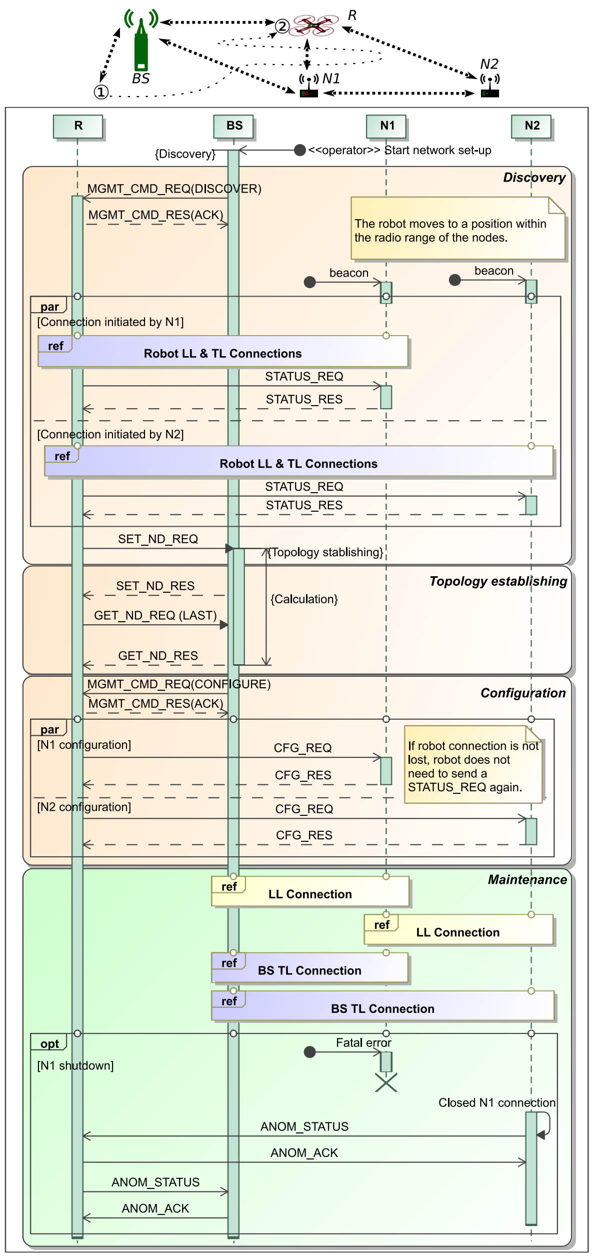

Fig. 5. Sequence diagram with SNSR operation.

VI. Implementation Details

SNSR was implemented as a flexible and highly configurable architecture easily adaptable to a wide variety of sensor networks. It only requires from the sensor network protocol stack the following specific functionalities: a) transmission of beacons in LL; b) security in TL and LL with handshakes that support authentication with certificates or that can apply security parameters negotiated in higher layers; c) a reliable TL; and d) possibility of adding a management application with access to the management plane in order to have control of the layers in the communication plane.

SNSR validation was performed with a simple and sensor network agnostic implementation that provided flexibility for testing different configurations, algorithms, and protocols but without reaching the low PHY and LL layers. The implemented SNSR uses an OSI-based protocol stack developed over an existing LL. The following LL were considered: a) IEEE 802.3 (*Ethernet*) and the wireless protocols of IEEE 802.11 (*Wi-Fi*), and b) tunneled LL over UDP. On top of LL, a LLAL (Link Layer Adaptation Layer), which supplies homogeneous functionalities to higher layers by meeting the above SNSR requirements, a NL, a TL, and an AL were added. The detailed description of these layers is omitted for brevity. NL uses a simple fixed 8-byte header with 16-bit addresses and does not support fragmentation or source routing. Local routing is static based on routing tables. In *Topology Establishing*, the minimum-hop routes between any origin-destination pair are computed. For robustness, several alternative routes are selected for each pair. TL uses a variable header between 1 and 7 bytes and adopts the same handshake and connection maintenance procedures as LL. It uses message-oriented connections with message segmentation and reassembly, and the same connection simultaneously carries one reliable and one unreliable data flow in each direction. The transport of the reliable flow adapts typical functionalities of other reliable protocols, e.g. TCP and QUIC [51]. The messages used in AL use *Protocol Buffers* [52] encoding. We adopt DTLS 1.2 [53] for secure connections. For maintainability, the same secure protocol was used in LL and TL.

SNSR was programmed in C/C++ under Linux due to efficiency and versatility. Two code libraries were used: protobuf and wolfSSL [54] to implement DTLS and cryptographic functions. The implementation was also virtualized in simulated scenarios using *NetEm* [55] and *WmediumD* [56] to emulate realistic communication channels. Robot navigation (e.g. planning and control) and node localization were implemented in ROS (Robotics Operating System) [57], a widely-used flexible open-source framework for robot software development.

VII. SNSR Evaluation

A. Security Analysis

Table VI shows the security of SNSR to the most common attacks, see e.g. [3], [4], [58], [59]. SNSR was analyzed with `SEC_LT` or `SEC_LTE` security modes, which provide encryption, authentication, and integrity protection in LL and TL. Attacks that involve characteristics that SNSR acquires from the adopted PHY and LL (e.g. *jamming* or *LL collision*

TABLE VI
SECURITY ANALYSIS SUMMARY

Target	**Attack**	**SNSR**	**[21]**	**[5]**	**[34]**
Network availability in PHY	*A1. Tampering*: physical manipulation of nodes [3]	⋈	×	×	⋈
	A2. Node destruction: total physical damage [59]	⋈	⋈	⋈	-
Network availability and service integrity in LL	*A3. Service request power attack*: prevention of sleep mode by sending many service requests [3]	✓	×	✓	-
	A4. Benign power attack: to compel victims to make power-intensive operations repeatedly [3]	✓	×	✓	-
Network availability and service integrity in NL and routing	*A5. Spoofed routing information*: transmission of false, altered, or repeated routing information [58]	✓	×	✓	-
	A6. Routing table overflow: to announce non-existent routes to fill routing tables and prevent the addition of new ones [58]	✓	×	✓	-
	A7. Routing table/cache poisoning: modifying stored routes by sending route updates to other nodes [58]	✓	×	✓	-
	A8. Black hole attack: to get the nodes to route their traffic through another node that discards it [58]	✓	×	✓	-
	A9. Sink hole attack: to divert traffic to the sink or other nodes to a malicious node [58]	✓	×	✓	-
	A10. Wormhole attack: to announce a tunnel between two nodes as the best route to a destination [58]	✓	×	✓	-
	A11. Acknowledgment spoofing: sending of forged acknowledgments [58]	✓	×	×	-
	A12. Rushing attack: to send/retransmit/alter packets quickly, getting them received before others [58]	✓	×	×	✓
	A13. Hello flooding: discovery message transmission to distant nodes to trigger failed attempts [3]	✓	×	×	×
	A14. Selective forwarding: forwarding some messages and discarding others in violation of the protocol [3]	⋈	⋈	⋈	-
	A15. Sybil: one node pretends to be many nodes [3]	⋈	⋈	⋈	⋈
	A16. Byzantine attack: coordination of several malicious nodes to perform other attacks in an amplified manner [58]	⋈	×	×	-
Network availability and service integrity in TL	*A17. TL flooding*: establishing many connections to exhaust the memory of the nodes [3]	✓	⋈	✓	-
	A18. TL desynchronization: forgery of control messages and sequence numbers to block connections [3]	✓	✓	✓	-
Privacy and secrecy	*A19. Eavesdropping*: the capture of transmitted information in a passive way [3]	✓	⋈	⋈	✓
	A20. Man-in-the-middle (MITM): to relay and alter the communication of two peers who are unaware of this [59]	✓	✓	×	✓
	A21. Attacks on cryptography: attacks on encryption algorithms, hash functions, or key exchange protocols [59]	✓	✓	⋈	×
	A22. Information disclosure: the sending of confidential data to unauthorized parties [58]	⋈	⋈	⋈	⋈
	A23. Traffic analysis: to analyze the traffic to infer its topology, critical nodes, and relationships, among others [3]	⋈	×	⋈	⋈
Data integrity	*A24. Packet injection*: false packets injection into the network [3]	✓	⋈	✓	×
	A25. Packet duplication: forwarding of previously captured packets [3]	✓	✓	✓	✓
	A26. Packet alteration: packet interception, modification, and resending [3]	⋈	⋈	⋈	⋈
	A27. Node replication: to capture a node and copy all its data to deploy replicas [3]	⋈	⋈	⋈	×

Legend: ✓attack-resistant, ⋈ largely reduces the effects, × non-resistant, - not considered

[3]) are not analyzed but, as described in Section V-F, they will be detected if they cause communication interruption. In addition, the security of SNSR is compared to that of: 1) ZigBee [50] using the *ZigBee Smart Energy* profile (version 1.2a) [21], used in installations that require a high level of security; 2) the INSENS [5] protocol using one BS, which covers many aspects of WSN security, allowing the analysis of all the considered attacks; and 3) work [34], a key management scheme using a UAV. *ZigBee Smart Energy* supports many configurations. We focus on the one most like SNSR: the use of centralized security with a trust center (TC) in a mesh network without periodic beacons using certificates signed by a CA and link keys preconfigured with the TC. For its analysis, in addition to the official documentation, previous studies have been used [60]–[62]. Work [34] does not consider many attacks, and the comparison is not fair. The same would happen with other related works focused on specific security aspects.

SNSR includes proactive and reactive security measures. First, R communicates directly with nodes to transmit critical data (e.g. configuration and routes), to obtain direct observations (e.g. identification, location, and status), and to collect the detected anomalies without intermediaries. SNSR is robust to *node replication* (*A27*) as nodes are localized and identified by R. If two replicas are close, the attack will only cause local communication problems. Attacks *A14* and *A26* require a previous attack to take control of a node. R communication with nodes or BS is direct and hence, immune to them. Node-BS or inter-node communications could be affected. Although SNSR does not contain specific mechanisms to prevent them, their effects are limited to the messages forwarded by the compromised nodes. Other security modules could be added to detect these attacks, see e.g. [63]. In case of *A26*, the packets will be discarded due to decryption error. It is not possible to add new nodes without prior authorization. A *Sybil A15* attack would be difficult to carry out: it would require the keys and certificates of several nodes that are authorized to occupy the same position. Moreover, its effects would be local.

Nodes do not have decision capability and only communicate with authorized entities. The implemented SNSR uses centrally-computed routes, being immune to *A5*, *A6*, *A7*, *A8*, *A9*, and *A10* attacks. Also, SNSR is naturally immune to *A3* attacks: first, entities discard messages from non-authorized entities; and second, failed attempts of connection with R are reported. Only pre-authorized incoming connections in TL are accepted, which naturally prevents *A17* attacks. SNSR naturally prevents *A16* as the operation of a node cannot be influenced by another node.

SNRS does not use shared keys. In *A1* attacks the most sensitive retrieved data are the entity keys, involving only local effects. Besides, nodes do not store data of the rest of the network, and the robot does not store private keys of other entities. A malicious node performing an *A22* attack would capture only the node keys and its accessible data.

Each entity detects anomalies and reports them to BS. Although *node destruction* (*A2*) attacks cannot be avoided, they are rapidly detected. Besides, in the implemented SNSR each node is configured with alternative routes, which largely reduces the effect of these attacks on routing. Also, *A4* attacks causing repeated connections/disconnections or sending incorrect broadcast messages are easily detected. *A12* and *A13* are usually classified as NL attacks, but can also be performed in other layers. The NL of the implemented SNSR does not offer services that can be attacked in these ways. Also, in LL a fake robot performing these attacks would cause nodes responding to fake beacons (*A13*) or connecting to a fake robot (*A12*). These unsuccessful connection establishment attempts will be reported as anomalies, and will not avoid connection with the real robot.

SNSR is flexible and upgradable with respect to the cryptographic techniques in LL and TL. Encryption prevents *A19* attacks. Authentication and CRL provides robustness to *A20* attacks. Also, message integrity and authentication in LL and TL prevents *A11*, *A18*, *A24*, *A25*, and *A26* attacks in these layers. Acknowledgements are not used in NL, preventing *A11* attacks in this layer. *Packet duplication* (*A25*) are prevented using sequence numbers. Finally, SNSR is not limited to specific cryptographic techniques. They can be updated in the event of a security breach, preventing *A21* attacks.

Performing complete *A23* attacks externally would require capturing all the network traffic. With an infiltrated node, only the source and destination addresses of the packets passing through it could be analyzed.

SNSR is resistant to many different attacks, and the few attacks that it is not immune to have limited/local effects and require a previous attack to obtain the keys of the compromised nodes and to replace one or more nodes. Moreover, SNSR was submitted to a *Red Team* analysis performed by network security experts. It was insensitive to false beacon attacks, BS DoS attacks, and robot LL exhaustion. It was very slightly affected by *node replication* and *node destruction*: both were detected as anomalies and in the first case, the replicated nodes were immediately revoked with no other degradation. During *jamming* attacks, communication was interrupted but the anomaly was immediately detected, and when *jamming* ended, SNSR continued to operate normally.

B. Performance Evaluation and Validation

SNSR has been extensively experimented in different scenarios using different types of robots. Nodes were implemented with *Raspberry Pi 1 Model B* equipped with WIFI dongles and, BS, with a laptop with a i7-6500U CPU with 2 cores, 4 GB RAM and 30 GB SSD HD. Nodes, R and BS are equipped with Ultra Wide Band (UWB) devices. R localizes the nodes and BS integrating UWB measurements using Particle Filters [64], which provided sufficiently low localization error indoors and outdoors. SNSR was configured with `SEC_LTE`. The DTLS cipher suite used was `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`. It uses ECC and AES, which is implemented in hardware in many platforms. Curve *prime256v1* was used in ECC (256 bits keys). Secure hash algorithm SHA-256 is used for MAC generation. Nodes sent monitoring measurements every 10 s. The maximum robot beacon transmission interval was 1 s. The packet hop limit was 18. Wi-Fi at 54 Mbps in ad-hoc mode was used as LL. LL MTU was fixed to 1232 bytes.

A series of 100 experiments was performed. The nodes were set in a 3x8 rectangular grid, with grid separation of 3 m, and BS was located at (0,0). In these experiments the robot was a *Pioneer 3AT* ground robot equipped with another *Raspberry Pi 1 Model B*. It followed pre-programmed trajectories at a speed of 0.6 m/s and included LIDAR self-localization using the AMCL algorithm [65]. Below, we present one experiment. Similar results were obtained in all the experiments performed.

First, *Discovery* was triggered. R took 75.23 s to automatically follow the trajectory of 43.33 m shown in Figure 6-top.

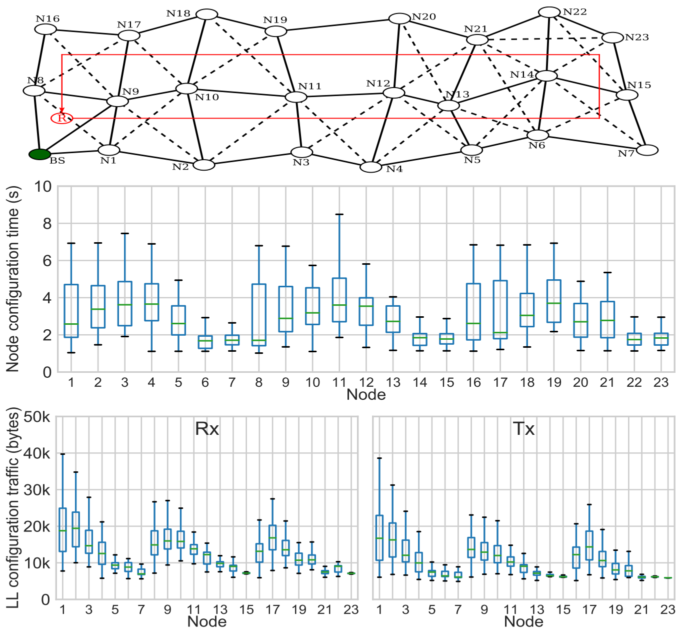

Fig. 6. Top) Discovered nodes and topology. Center) Time between reception of configuration request and the first BS measurement ACK for each node. Bottom) Received and transmitted bytes in LL until each node received the first measurement ACK from BS.

R discovered the nodes and located them with an average error of 0.83 m, see Figure 6-top. In that setting the measured node communication range was 3.5 m. We adopt a conservative node neighbor criterion distance of r=5 m, such that most actual neighbor nodes were assigned as neighbors even with the node localization error. The resulting neighbor connectivity is shown with lines (solid and dashed) in Figure 6-top. This criterion wrongly assigned as neighbors some nodes which distance was actually higher than the communication range. These errors will be later detected as anomalies in *Maintenance*, and corrected. Next, R computed the routes and created the new NTDB in 1.68 ms, and sent it to BS. In average nodes required 4.09 hops to reach BS.

In *Configuration* R followed the same trajectory. Each node measured the time between it received the configuration request and it received the first measurement ACK from BS –for short, configuration time. Figure 6-center shows the configuration times in the 100 performed experiments (percentile 25, 75, and median). The average times were between 1.5 and 3.8 s, with an average of 2.96 s. At the robot initial location, the robot configured several nodes simultaneously, causing collisions and requiring several connection establishment retries, resulting in larger configuration times. In all the performed experiments the configuration times were always below 10 s. The transmitted and received bytes in LL by each node until they receive the first measurement ACK is shown in Figure 6-bottom. In average each node received 12.50 Kbytes and transmitted, 10.25 Kbytes. Traffic is higher for nodes near BS due to packet forwarding. In all the performed experiments LL traffic was never above 46 Kbytes. Secure handshaking was responsible for a high part of the traffic, each handshake in nodes took in average 0.97 s and required ∼1,100 bytes in each direction (60% of them due to certificates). A node performs

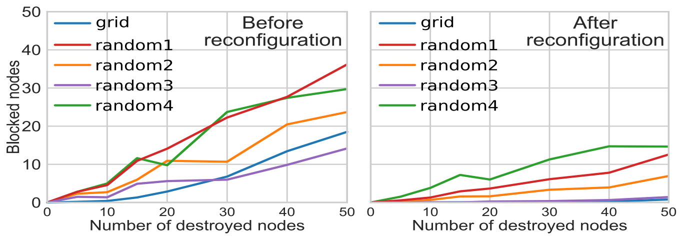

Fig. 7. Node destruction effects before (left) and after (right) reconfiguration.

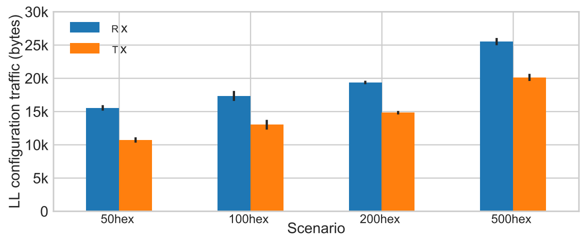

Fig. 8. Average node LL traffic during configuration in scenarios with 50, 100, 200 and 500 nodes.

handshakes only with R, BS and, its neighbors, involving low configuration times and traffic even in large networks.

Next, *Maintenance* was performed. Each node tried to connect to BS, R, and its authorized neighbors, but several inter-node connections could not be established (shown as dashed lines in Figure 6-top). They were detected as anomalies, and it was decided to perform a new *Discovery*. In the next *Topology Establishing* stage, the connectivity anomalies and node status were used to correct the topology. The updated topology, shown with solid lines in Figure 6-top, contained no errors. In the next stages, routes were recomputed and nodes, reconfigured, and no connection errors were detected. A non-restrictive criterion was used to assign neighbors to exploit SNSR self-correction. Many security methods obtain topological information through node interaction, which is prone to attacks. SNSR uses a simple criterion based on node location (avoiding node interaction and hence, improving security in network set-up stages), and later corrects topology errors and self-adapts, using the robot.

For further validation we performed virtualized experiments where SNSR was submitted to massive node destruction attacks: networks with 200 nodes with different topologies in which different numbers of randomly selected nodes were simultaneously destroyed. A total of 50 experiments of each case and topology were performed. Figure 7-left shows the total number of blocked nodes –disconnected from BS– in each topology, some of them were physically disconnected and in others, the routes to BS were damaged. All blocked nodes were detected as anomalies. Next, SNSR performed *Discovery*, *Topology Establishing*, and *Configuration* to provide the nodes with new routing information. Figure 7-right shows the total number of blocked nodes after reconfiguration. Only the physically-blocked nodes kept disconnected. SNSR largely reduced the impact of node destruction in all scenarios even in the unlikely case of having 50 nodes simultaneously destroyed.

SNSR was tested in realistic applications, such as bridge and cement kiln monitoring, see Figure 1. The used nodes, BS and SNSR configuration were the same as in the above experiments. R in the bridge scenario was based on a *DJI Flamewheel F450* platform endowed with a *PixRacer* autopilot and a low-cost *Khadas VIM3* for processing and logging. R used in the cement kiln was a custom-design hexarotor with a 3D-LiDAR for GNSS-free self-localization and an *Intel NUC* computer. SNSR was validated in more than 20 experiments in each scenario. The data from one experiment in each scenario is provided as additional material [15].

C. Sensitivity Analysis

Next, the sensitivity to topology and network size is analyzed using the SNSR virtualization. First, 100 different scenarios were designed in an 210 m^2 area with 48 nodes and one BS, all at random locations but forcing network connectivity. R followed a zig-zag pattern trajectory that ensured communication with BS and every node along its trajectory. The resulting mean node configuration times were between 2.3 and 9.1 s depending on the scenario, showing moderate sensitivity to topology. The cases with longer configuration times occurred when nodes distant from BS were configured before existing a path of already configured nodes to reach it. The LL traffic also showed low topology dependence. The cases with higher traffic were caused by temporary loops in the configuration transitory period due to the node configuration order. In average each node received 19,730 bytes and transmitted 16,190 bytes. In the scalability analysis, four scenarios were designed with 50, 100, 200, and 500 nodes distributed in hexagonal grid with constant inter-node distance and the BS located at the center of the setting. The mean number of neighbors in each scenario were 4.88, 5.18, 5.43, and 5.64, respectively. The mean number of hops to BS were 2.86, 4.02, 5.63, and 8.83. The resulting LL traffic during configuration averaging 100 experiments for each scenario is shown in Figure 8, evidencing good scalability. This slight traffic increment is caused by the higher number of neighbors (more handshakes) and hops to BS (more packet forwarding).

VIII. Comparison with Related Works

It is not easy to find works to compare SNSR. SNSR considers the overall security of WSN (from the network set-up), whereas most of the works focus on specific security aspects. SNSR uses a robot in WSN security, which has only been considered theoretically in few papers. Table I supplies a first comparison to the papers covered in the related work, including those using robots. Additionally, Table VI includes a detailed comparison to [5], [21], and [34] against various attacks, concluding that SNSR resists more attacks. This section complements the above comparisons.

The application of PKC and ECC in WSN has already been performed by [16]–[21], and there are different approaches to authenticate public keys [10], [20], [22], [23]. Our architecture could work with any of them, but we propose PKI with session resumption and certificate pre-validation for efficiency. Although SNSR requires an authenticated key exchange protocol, it is flexible and does not impose a specific one. In SNSR implementation, we prefer to use the standard and well-proven DTLS protocol.

In SNSR, all entities detect anomalies (unlike [25] and [26]), according to their role, in line with [27], but in a much simpler way: reporting communication failures or behaviors that do not follow the specification (without added computational load) and taking advantage of the direct observation of the robot. Anomaly detection in nodes mainly involves transmission costs but provides faster attack detection and avoids the continuous use of the robot. Anomaly collection and analysis are centralized in the BS (unlike [28] and [8]) to free resources on the nodes. The simplification provided by SNSR makes it possible to implement a specification-based IDS and does not exclude the use of other IDSs such as [38] and [9] or detectors for specific attacks using the provided infrastructure. Unlike many IDSs that only propose detection algorithms [24], [28], [29], SNSR also has mitigation mechanisms that can change the network during attacks (e.g., urgent CRL update and direct configuration). Only [25] and [8] propose other mitigation mechanisms. The rest simply warn the operator (SNSR does so in some cases, not always) or do not consider mitigation.

SNSR significantly reduces the tasks performed by the nodes, the weakest component in WSN security, with the computational and energy savings that this entails. For example, nodes do not perform neighbor or BS discovery, a common task in other works [5], [21], [34]. Additionally, SNSR avoids requiring dynamic routing protocols, further reducing node influence. It can support various routing mechanisms. It could also use static routing during network creation and later change to a distinct routing such as SDN.

Some works assume that the security requirement during the initial network set-up may be lower than during operation because there is more supervision. For example, in [33] at the beginning, all nodes share the same initial key. However, [5] and SNSR consider in their design that the nodes could have preinstalled malware or there could always be hidden sniffers.

Regarding WSN security with robots, work [31] presents a method for CH election assuming that the malicious nodes report higher energy consumption and cannot deceive the UAV. Work [32] requires the robot to have the same sensing capabilities as the sensor nodes to determine node trustworthiness. In these works security is a secondary objective and focus only on particular problems. Work [33] distributes symmetric keys to a group of nodes, but an attacker could listen to the transmitted keys if he knows a global key preinstalled on all nodes. Also, the robot is authenticated using hash chain variants, which are susceptible to rushing or packet duplication attacks. Work [34] proposes a key agreement protocol using PKC with public keys obtained from a UAV to avoid storing them in nodes. Neither [33] nor [34] considers the capture of the UAV. This would be fatal to security as the UAV has irreplaceable information, which is also preconfigured in the nodes. In SNSR, the robot does not distribute keys. It pre-validates public keys, so nodes need less effort to detect incorrect keys. In addition, the capture of a robot is not critical since it is possible to revoke the certificate of the captured robot and use another robot with different certificate and keys.

IX. Conclusions

This paper proposes SNSR, a novel, open, and flexible security architecture that exploits robot-sensor network cooperation to achieve high security levels without using complex mechanisms. Its operation is structured in stages organized in a feedback approach, enabling repeating them to adapt to network changes, attacks, or correct errors. SNSR benefits from the actuation capability of the robot, which locates and authenticates nodes, interacts with nodes directly without intermediaries to send them configurations and receive status and anomaly reports, and can also be commanded to obtain direct observations to confirm/discard detected anomalies. These uses of the robot provide SNSR with enhanced attack detection and mitigation capabilities (see Appendix), and improves prevention by replacing tasks that in traditional architectures are prone to attacks or require preset information. It has been extensively experimented in indoor and outdoor scenarios evidencing its security, efficiency, and scalability.

The extension of SNSR with several cooperating robots and base stations (for larger scenarios) is object of current research.

Appendix

Next, the security advantages of SNSR are analyzed. Consider a network of N static nodes with a security architecture in which the anomalies detected by all entities are collected at BS, where they are analyzed to detect attacks and decide the best actions/reconfigurations to mitigate their effects. Let S be the set of sensor nodes in the network, and S_1, the nodes that are directly connected to BS. The sensor network status at time k is represented by state vector $\mathbf{x_k}$, and the sensor network performance, by $J(\mathbf{x_k})$. Vector $\mathbf{y_{j,k}}$ represents one anomaly detected/observed by node j at time k. The anomaly observation model is $\mathbf{y_{j,k}}$=$g_{j,k}(\mathbf{x_k})$. Observation is performed under Gaussian noise, i.e. anomalies are modelled as Gaussian multivariate random variables. The effect of the potential attacks when sending $\mathbf{y_{j,k}}$ to BS are modeled by function $hy_{j,k}$. BS analyzes $\{hy_{j,k}(\mathbf{y_{j,k}})\}$, the set of all received anomalies; if an attack is detected, it selects $\mathbf{u_{i,k}}$, the actions/configurations for every node i, that maximize the network performance improvement after the attack, $\mathbf{u_{i,k}} = \max\limits_{\mathbf{u_{i,k}}}(J(\mathbf{x_{k+1}}) - J(\mathbf{x_k}))$. $hu_{i,k}$ models the effect of attacks when sending $\mathbf{u_{i,k}}$ to node i. When the nodes execute their actions, their behaviour change: the state at k+1 is $\mathbf{x_{k+1}}$. The effect of attacks on anomaly collection is as follows. Some attacks, e.g., malicious nodes, can send fake anomalies. Others, e.g., DoS, can reduce the rate of successful anomaly transmission. As described in Section V-B, connections between entities are secure: the effects of packet injection/alteration/replication attacks are negligible. Similarly to [66], we model the data received at BS as:

$$\hat{\mathbf{y}}_{\mathbf{j,k}} = hy_{j,k}(\mathbf{y_{j,k}}) = \gamma_{j,k}(\mathbf{y_{j,k}} + \bar{\mathbf{y}}_{\mathbf{j,k}}), \tag{14}$$

where $\gamma_{j,k}$ takes value on 1 or 0, and $\bar{\mathbf{y}}_{\mathbf{j,k}}$ is a fake anomaly from a malicious node. Actions are sent by BS using a secure connection: they are authenticated and are robust to malicious nodes sending fake actions. The data received by node i is:

$$\hat{\mathbf{u}}_{\mathbf{i,k}} = hu_{i,k}(\mathbf{u_{i,k}}) = \delta_{i,k}\mathbf{u_{i,k}}, \tag{15}$$

where $\delta_{i,k}$ takes value on 1 or 0.

The above is used to compare SNSR to a traditional centralized architecture with no robot (TC-WSN). Both use the same protocols and mechanisms. In both, BS collects directly the anomalies from, and delivers the actions to, every node $j{\in}S_1$. The only difference is that SNSR uses R to collect the anomalies from (and deliver the actions to) every node $j{\in}S \setminus S_1$.

Proposition 1. *SNSR collects more attack information than TC-WSN*

Proof. We adopt the widely-used Fisher information of random variables to measure the information of anomalies [67]. Using Eq. (14), the information received at BS from $\mathbf{y_{j,k}}$ is $(py_{j,k}I_{j,k})$, where $I_{j,k}$ is the Fisher information of $(\mathbf{y_{j,k}}{+}\bar{\mathbf{y}}_{\mathbf{j,k}})$, and $py_{j,k}$ is the probability that $(\mathbf{y_{j,k}}{+}\bar{\mathbf{y}}_{\mathbf{j,k}})$ is received at BS. Anomalies from any node $j{\in}S_1$ are sent directly to BS. Assuming that all single-hop transmissions have the same success ratio $p{\in}[0, 1]$, we have $py_{j,k}{=}p \; \forall j{\in}S_1$. In TC-WSN, collecting at BS anomalies from node $j{\in}S \setminus S_1$ through the network requires $ny_j \geq 2$ transmissions: the success ratio is $py_{j,k}{=}p^{ny_j}$. $\mathbf{y_{B,k}}$ is an anomaly detected by BS. Its Fisher information is $I(\mathbf{y_{B,k}})$. The Fisher information is additive [67]. In TC-WSN, the information of all the anomalies collected at BS is:

$$I_k = \sum_{y_{B,k}} I(\mathbf{y_{B,k}}) + \sum_{j\in S_1} \sum_{y_{j,k}} pI_{j,k} + \sum_{j\in S\setminus S_1} \sum_{y_{j,k}} p^{ny_j} I_{j,k} \quad (16)$$

where $\sum_{y_{B,k}}$ and $\sum_{y_{j,k}}$ refer respectively to all the anomalies detected by BS and by node j.

In SNSR, $\mathbf{y_{R,k}}$ is one anomaly detected by R. Its Fisher information is $I(\mathbf{y_{R,k}})$. Collecting at BS anomalies from node $j{\in}S \setminus S_1$ using R requires two transmissions. Hence, in SNSR the information of all the anomalies collected at BS is:

$$I'_k = \sum_{y_{R,k}} I(\mathbf{y_{R,k}}) + \sum_{y_{B,k}} I(\mathbf{y_{B,k}}) + \sum_{j\in S_1} \sum_{y_{j,k}} pI_{j,k} + \sum_{j\in S\setminus S_1} \sum_{y_{j,k}} p^2 I_{j,k} \quad (17)$$

The Fisher information is non negative [67]. Since $ny_j{\geq}2$, it is guaranteed that $I'_k{>}I_k$ for any network topology. □

Proposition 2. *SNSR delivers actions/reconfigurations with non-lower success probability than TC-WSN*

Proof. In TC-WSN, sending $\mathbf{u_{i,k}}$ to any node $i{\in}S{\setminus}S_1$ requires $nu_i \geq 2$ transmissions: the successful transmission ratio is $pu_i{=}p^{nu_i}$. In SNSR, actions $\mathbf{u_{i,k}} \; \forall i \in S \setminus S_1$ are sent using R, hence the success ratio is $pu'_i{=}p^2$. Also, in SNSR and in TC-WSN, BS transmits actions to any node $i{\in}S1$ directly and hence with successful transmission ratios $pu'_i{=}pu_i{=}p$. Since $nu_i{\geq}2$, it is guaranteed that $pu'_i \geq pu_i \; \forall i$ and hence, $\delta'_{i,k} \forall i$ takes value on 1 with a frequency non-lower than $\delta_{i,k}$. □

Proposition 3. *SNSR recovers from an attack faster than TC-WSN*

Proof. The network performance at $k{+}1$ can be obtained from $J(\mathbf{x_k})$ using the actions $\mathbf{u_{i,k}}$ received by nodes at time k as:

$$J(\mathbf{x_{k+1}}) \simeq J(\mathbf{x_k}) + \sum_{i\in S} \frac{\partial J(\mathbf{x_k})}{\partial \mathbf{u_{i,k}}} \delta_{i,k} \mathbf{u_{i,k}}, \quad (18)$$

where $\frac{\partial J(\mathbf{x_k})}{\partial \mathbf{u_{i,k}}} \mathbf{u_{i,k}}$ represents the performance change caused by action $\mathbf{u_{i,k}}$, and $\delta_{i,k}$ was defined in (15). The performance increments in SNSR and TC-WSN are respectively:

$$\Delta J' = \sum_{i\in S} \delta'_{i,k} \frac{\partial J(\mathbf{x_k})}{\partial \mathbf{u'_{i,k}}} \mathbf{u'_{i,k}}, \quad \Delta J = \sum_{i\in S} \delta_{i,k} \frac{\partial J(\mathbf{x_k})}{\partial \mathbf{u_{i,k}}} \mathbf{u_{i,k}} \quad (19)$$

As stated before, the actions to recover from an attack are selected to maximize the network performance improvement. From Prop. (1), SNSR collects more attack information, and hence, will tend to select actions that will cause larger network performance improvements, i.e. $\frac{\partial J(\mathbf{x_k})}{\partial \mathbf{u'_{i,k}}} \mathbf{u'_{i,k}} > \frac{\partial J(\mathbf{x_k})}{\partial \mathbf{u_{i,k}}} \mathbf{u_{i,k}}$. Additionally, from Prop. (2), $\delta'_{i,k}$ takes value on 1 with a frequency non-lower than $\delta_{i,k}$. Hence, it is guaranteed that $\Delta J'{>}\Delta J$: after an attack the increment in network performance is higher in SNSR than in TC-WSN. □

References

[1] D. Kandris, C. Nakas, D. Vomvas, and G. Koulouras, "Applications of Wireless Sensor Networks: An Up-to-Date Survey," *Applied System Innovation*, vol. 3, no. 1, p. 14, feb 2020.

[2] J. Y. Yu, E. Lee, S. R. Oh, Y. D. Seo, and Y. G. Kim, "A Survey on Security Requirements for WSNs: Focusing on the Characteristics Related to Security," *IEEE Access*, vol. 8, pp. 45 304–45 324, 2020.

[3] R. Di Pietro, S. Guarino, N. V. Verde, and J. Domingo-Ferrer, "Security in wireless ad-hoc networks - A survey," *Computer Communications*, vol. 51, pp. 1–20, 2014.

[4] I. Tomic and J. A. McCann, "A Survey of Potential Security Issues in Existing Wireless Sensor Network Protocols," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1910–1923, dec 2017.

[5] J. Deng, R. Han, and S. Mishra, "INSENS: Intrusion-tolerant routing for wireless sensor networks," *Computer Communications*, vol. 29, no. 2, pp. 216–230, jan 2006.

[6] O. R. Ahutu and H. El-Ocla, "Centralized Routing Protocol for Detecting Wormhole Attacks in Wireless Sensor Networks," *IEEE Access*, vol. 8, pp. 63 270–63 282, 2020.

[7] G. A. N. Segura, S. Skaperas, A. Chorti, L. Mamatas, and C. B. Margi, "Denial of service attacks detection in software-defined wireless sensor networks," in *IEEE Int Conf on Comms. Workshops*, 2020, pp. 1–7.

[8] C. Miranda, G. Kaddoum, E. Bou-Harb, S. Garg, and K. Kaur, "A Collaborative Security Framework for Software-Defined Wireless Sensor Networks," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2602–2615, 2020.

[9] S. Jiang, J. Zhao, and X. Xu, "SLGBM: An Intrusion Detection Mechanism for Wireless Sensor Networks in Smart Environments," *IEEE Access*, vol. 8, pp. 169 548–169 558, sep 2020.

[10] E. Yuan, L. Wang, S. Cheng, N. Ao, and Q. Guo, "A Key Management Scheme Based on Pairing-Free Identity Based Digital Signature Algorithm for Heterogeneous Wireless Sensor Networks," *Sensors*, vol. 20, no. 6, p. 1543, mar 2020.

[11] C. Meshram, C. C. Lee, S. G. Meshram, and A. Meshram, "OOS-SSS: An Efficient Online/Offline Subtree-Based Short Signature Scheme Using Chebyshev Chaotic Maps for Wireless Sensor Network," *IEEE Access*, vol. 8, pp. 80 063–80 073, 2020.

[12] A. Ollero et al., "The AEROARMS Project: Aerial robots with advanced manipulation capabilities for inspection and maintenance," *IEEE Robotics Automation Magazine*, vol. 25, no. 4, pp. 12–23, 2018.

[13] S. Shue and J. M. Conrad, "A survey of robotic applications in wireless sensor networks," in *Proc. IEEE Southeastcon*, 2013, pp. 1–5.

[14] Popescu, Stoican, Stamatescu, Chenaru, and Ichim, "A Survey of Collaborative UAV–WSN Systems for Efficient Monitoring," *Sensors*, vol. 19, no. 21, p. 4690, oct 2019.

[15] F. J. Fernández-Jiménez and J. R. Martínez-de Dios, "Additional material of A Robot-Sensor Network Security Architecture for Monitoring Applications," 2020. [Online]. Available: http://dx.doi.org/10.21227/62fj-8b04

[16] D. Kim and S. An, "Efficient and Scalable Public Key Infrastructure for Wireless Sensor Networks," in *The 2014 International Symposium on Networks, Computers and Communications*. IEEE, jun 2014, pp. 1–5.

[17] A. Toubal, B. Bengherbia, M. O. Zmirli, and A. Guessoum, "Fpga implementation of a wireless sensor node with built-in security coprocessors for secured key exchange and data transfer," *Measurement: Journal of the Int. Measurement Confederation*, vol. 153, p. 107429, 3 2020.

[18] R. Qazi, K. N. Qureshi, F. Bashir, N. U. Islam, S. Iqbal, and A. Arshad, "Security protocol using elliptic curve cryptography algorithm for wireless sensor networks," *Journal of Ambient Intelligence and Humanized Computing*, vol. 1, p. 3, 4 2020.

[19] S. M. Hussein, J. A. L. Ramos, and J. A. Álvarez Bermejo, "Distributed key management to secure iot wireless sensor networks in smart-agro," *Sensors*, vol. 20, p. 2242, 4 2020.

[20] J. Liu, L. Wang, and Y. Yu, "Improved security of a pairing-free certificateless aggregate signature in healthcare wireless medical sensor networks," *IEEE Internet of Things Journal*, vol. 7, pp. 5256–5266, 6 2020.

[21] ZigBee Alliance, "ZigBee Smart Energy Standard," pp. 1–628, 2014.

[22] R. Hummen, J. H. Ziegeldorf, H. Shafagh, S. Raza, and K. Wehrle, "Towards viable certificate-based authentication for the Internet of Things," in *Proc. 2013 ACM Workshop on Hot Topics on Wireless Network Security and Privacy*. ACM Press, 2013, pp. 37–41.

[23] D. Abbasinezhad-Mood and M. Nikooghadam, "An anonymous ecc-based self-certified key distribution scheme for the smart grid," *IEEE Transactions on Industrial Electronics*, vol. 65, pp. 7996–8004, 10 2018.

[24] M. Trnka, J. Svacina, T. Cerny, E. Song, J. Hong, and M. Bures, "Securing internet of things devices using the network context," *IEEE Transactions on Industrial Informatics*, vol. 16, pp. 4017–4027, 6 2020.

[25] D. B.D. and F. Al-Turjman, "A hybrid secure routing and monitoring mechanism in IoT-based wireless sensor networks," *Ad Hoc Networks*, vol. 97, p. 102022, feb 2020.

[26] F. Hidoussi, H. Toral-Cruz, D. E. Boubiche, K. Lakhtaria, A. Mihovska, M. Voznak, and M. V. voznak, "Centralized ids based on misuse detection for cluster-based wireless sensors networks," *Wireless Pers Comms.*, vol. 85, pp. 207–224, 2015.

[27] S. S. Wang, K. Q. Yan, S. C. Wang, and C. W. Liu, "An integrated intrusion detection system for cluster-based wireless sensor networks," *Expert Systems with Applications*, vol. 38, pp. 15 234–15 243, 11 2011.

[28] F. Li, R. Xie, Z. Wang, L. Guo, J. Ye, P. Ma, and W. Song, "Online Distributed IoT Security Monitoring With Multidimensional Streaming Big Data," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4387–4394, may 2020.

[29] C. Hongsong, M. Caixia, F. Zhongchuan, and C. H. Lee, "Novel ldos attack detection by spark-assisted correlation analysis approach in wireless sensor network," *IET Information Security*, vol. 14, pp. 452–458, 7 2020.

[30] Z. Teng, B. Pang, C. Du, and Z. Li, "Malicious Node Identification Strategy with Environmental Parameters," *IEEE Access*, vol. 8, pp. 149 522–149 530, 2020.

[31] G. Wang, B. Lee, J. Ahn, and G. Cho, "A UAV-assisted CH election framework for secure data collection in wireless sensor networks," *Future Generation Computer Systems*, vol. 102, pp. 152–162, jan 2020.

[32] B. Jiang, G. Huang, T. Wang, J. Gui, and X. Zhu, "Trust based energy efficient data collection with unmanned aerial vehicle in edge network," *Transactions on Emerging Telecommunications Technologies*, mar 2020.

[33] B. Tas and A. S. Tosun, "Mobile assisted key distribution in wireless sensor networks," in *IEEE Int. Conference on Communications*, 2011.

[34] O. K. Sahingoz, "Large scale wireless sensor networks with multi-level dynamic key management scheme," *Journal of Systems Architecture*, vol. 59, no. 9, pp. 801–807, oct 2013.

[35] A. K. Gautam and R. Kumar, "A comprehensive study on key management, authentication and trust management techniques in wireless sensor networks," *SN Applied Sciences*, vol. 3, p. 50, 1 2021.

[36] A. M. Hegland, E. Winjum, S. F. Mjolsnes, C. Rong, O. Kure, and P. Spilling, "A survey of key management in ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 3, pp. 48–66, 2006.

[37] M. Numan, F. Subhan, W. Z. Khan, S. Hakak, S. Haider, G. T. Reddy, A. Jolfaei, and M. Alazab, "A systematic review on clone node detection in static wireless sensor networks," *IEEE Access*, vol. 8, pp. 65 450–65 461, 2020.

[38] S. Pundir, M. Wazid, D. P. Singh, A. K. Das, J. J. P. C. Rodrigues, and Y. Park, "Intrusion Detection Protocols in Wireless Sensor Networks Integrated to Internet of Things Deployment: Survey and Future Challenges," *IEEE Access*, vol. 8, pp. 3343–3363, 2020.

[39] M. Ndiaye, G. P. Hancke, and A. M. Abu-Mahfouz, "Software Defined Networking for Improved Wireless Sensor Network Management: A Survey." *Sensors (Basel, Switzerland)*, vol. 17, no. 5, may 2017.

[40] A. Wichmann, B. D. Okkalioglu, and T. Korkmaz, "The integration of mobile (tele) robotics and wireless sensor networks: A survey," *Computer Communications*, vol. 51, pp. 21–35, 9 2014.

[41] F. Hu, Q. Hao, and K. Bao, "A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[42] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, "SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, apr 2015, pp. 513–521.

[43] F. Olivier, G. Carlos, and N. Florent, "SDN Based Architecture for Clustered WSN," in *IEEE Int. Conf. on Innovative Mobile and Internet Services in Ubiquitous Computing*, jul 2015, pp. 342–347.

[44] H. Huang and A. V. Savkin, "Viable path planning for data collection robots in a sensing field with obstacles," *Computer Communications*, vol. 111, pp. 84 – 96, 2017.

[45] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 266–282, 2014.

[46] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software*, vol. 11, no. 1, pp. 37–57, 1985.

[47] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. on Information Theory*, vol. 22, no. 6, pp. 644–654, nov 1976.

[48] S. Law, Laurie and Menezes, Alfred and Qu, Minghua and Solinas, Jerry and Vanstone, "An efficient protocol for authenticated key agreement," *Designs, Codes and Cryptography*, vol. 28, pp. 119–134, 2003.

[49] Certicom Research, "Standards for Efficient Cryptography 1 (SEC 1): Elliptic Curve Cryptography," *Standards for Efficient Cryptography*, vol. 1, no. Sec 1, pp. 1–22, 2009.

[50] ZigBee Alliance, "Zigbee Specification," *Zigbee Alliance website*, 2015.

[51] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-transport-22, Jul. 2019, work in Progress.

[52] Google Inc., "Protocol Buffers," accessed: 2019-04-29. [Online]. Available: https://developers.google.com/protocol-buffers/

[53] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security Version 1.2," Internet Requests for Comments, RFC 6347, 2012.

[54] wolfSSL Inc., "wolfSSL Embedded SSL/TLS Library," accessed: 2019-04-27. [Online]. Available: https://www.wolfssl.com/products/wolfssl/

[55] S. Hemminger and Others, "Network emulation with NetEm," in *Linux conf au*, 2005, pp. 18–23.

[56] cozybit Inc., "cozybit/wmediumd: mac80211_hwsim modifications and related stuff," accessed: 2019-08-04. [Online]. Available: https://github.com/cozybit/wmediumd

[57] Open Robotics, "Robot Operating System (ROS)," accessed: 2019-08-04. [Online]. Available: https://www.ros.org/

[58] J. Sen, "Routing Security Issues in Wireless Sensor Networks: Attacks and Defenses," *Sustainable Wireless Sensor Networks*, pp. 279–309, 2010.

[59] J. Deogirikar and A. Vidhate, "Security attacks in IoT: A survey," *Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017*, pp. 32–37, 10 2017.

[60] E. Yüksel, "Analysing ZigBee Key Establishment Protocols," 2012.

[61] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards," *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2007.

[62] D. M. Shila, X. Cao, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen, "Ghost-in-the-Wireless: Energy Depletion Attack on ZigBee," no. Mic, pp. 1–13, 2014.

[63] T. H. Hai and E. Huh, "Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge," in *IEEE Int. Symp. on Network Comp. and Applications*, July 2008, pp. 325–331.

[64] J. R. Martínez-de Dios and et al., "Aerial robot coworkers for autonomous localization of missing tools in manufacturing plants," in *Intl. Conf. on Unmanned Aircraft Systems*, 2018, pp. 1063–1069.

[65] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[66] D. Wang, Z. Wang, B. Shen, F. E. Alsaadi, and T. Hayat, "Recent advances on filtering and control for cyber-physical systems under security and resource constraints," *Journal of the Franklin Institute*, vol. 353, no. 11, pp. 2451 – 2466, 2016.

[67] Z. Pablo, "Fisher information properties," *Entropy*, vol. 17, pp. 4918–4939, 7 2015.