EAI Endorsed Transactions
on Ubiquitous Environments Research Article **ICST.ORG**

A Middleware Approach for IEEE 802.15.4 Wireless Sensor Networks Security

S. Marchesani[1], L. Pomante[1,*], M. Pugliese[1] and F. Santucci[1]

[1]University of L'Aquila – DEWS (ITALY)

Abstract

Recent years have seen the growth of interest for middleware (MW) approaches in resource-constrained systems like Wireless Sensor Networks (WSNs). In this context, available MW platforms usually provide an Application Layer with different basic services, but do not provide security services such as secure links management or intrusion detection. Nevertheless, since WSN applications normally require the collection of reliable data from the sensing units, secure communications should be guaranteed even in the presence of resource constraints. So, this paper present a novel MW approach directly tailored to an IEEE 802.15.4-based WSN. The security-related components of the proposed MW include a light yet powerful cryptographic scheme (TAKS) and an Intrusion Detection System (WIDS). According to the performance and overhead assessment, we may argue that the proposed architecture has potentials to be exploited in realistic application scenarios that rely on the IEEE 802.15.4.

Keywords: security, middleware, wireless sensor networks, cryptographic scheme, intrusion detection system.

Received on 27 April 2015, accepted on 28 May 2015, published on 13 July 2015

doi: 10.4108/ue.2.5.e1

1. Introduction

In the last years, the so called *Wireless Sensor Networks* (WSNs) has attracted a lot of attentions in both academic and industrial domains. This is mainly due to the fact that WSNs could represent the infrastructure able to support the *Internet of Things* (IoT) architecture. In fact, a WSN is a distributed system, composed of a set of *sensor nodes* and a *sink node* able to communicate by means of wireless channels, that could be used for pervasive and ubiquitous data collection and processing. For this, WSNs have already found application in several areas, such as environmental monitoring, health monitoring, intelligent building, intelligent transportations, and so on, e.g. in [2]. In the depicted context, that is typically resource-constrained, particular attention has been devoted to development of *middleware platforms*. A middleware is a software platform used to hide complexity and heterogeneity of the underlying physical platform and network and to offer several services to the *Application Layer*, eventually providing an application execution environment [26]. When the underlying physical network is a WSN, considering typical monitoring oriented applications, data and system reliability are also required. Although security is not usually included in the services portfolio provided by middleware platforms for WSNs, reliability involves security issues: so, a middleware for WSNs should not ignore aspects such as secure data transmission and intrusion detection.

This paper deals with the definition and development of a new middleware framework to provide security in WSNs: in particular, a reference architecture for the middleware is proposed and the main design choices are discussed. Moving from our previous work, we focus on a hybrid

*Corresponding author. E-mail: luigi.pomante@univaq.it

cryptographic scheme called TAKS and an *Intrusion Detection System* (IDS) based on a simplified version of *Hidden Markov Models* (HMMs) called *Weak Process Models* (WPMs). The relevant feature of the presented work is related to the fact that the proposed architecture is tailored to real-world IEEE 802.15.4-based WSNs.
The remainder of this paper is organized as follow: Section 2 deals with background and motivations that have led us to propose a new middleware and also state-of-art about middleware platforms for WSNs that handle security is reported. Section 3 deals with the provision of security services for IEEE 802.15.4 networks and Section 4 with the proposed middleware architecture. Section 5 and 6 are then focused on the secure transmission service (which refers to a WSN-oriented cryptographic scheme), the intrusion detection service and, specifically, the issues related to implementation on the protocol stack. Finally, Section 7 reports our validation approach while in Section 8 some conclusive comments and future works are reported as well.

2. Background and motivations

Usually, WSNs are used in monitoring and control applications wherein system resources are very constrained. In particular, sensor nodes are often battery powered and, once deployed, not easily accessible by humans. So, the energy consumption should be carefully optimized. Moreover, a WSN should be flexible especially with respect to node heterogeneity. In fact, the burden of computation may vary from node to node and the exploitation of a heterogeneous network setup could be beneficial.
IEEE 802.15.4 has been designed to achieve these goals. It is a standard which specifies the *Physical Layer* and *Media Access Control* (MAC) for low-rate wireless personal area networks (LR-WPANs) [5]. A LR-WPAN is a simple, low-cost communication network that allows wireless connectivity in applications with limited power and relaxed throughput requirements. The main objectives of a LR-WPAN are easy installation, reliable data transfer, short-range operation, extremely low cost, and a reasonable battery life, while maintaining a simple and flexible protocol. Sensor nodes produced by different manufacturers, but following the Physical Layer specification, would be able to correctly communicate. Therefore, the exploitation of this standard implicitly gives the capability to manage heterogeneous WSNs. Moreover, IEEE 802.15.4 provides to the higher layers two types of services: the *MAC data service* and the *MAC management service*. The former one provides services to exchange data in the network, while the latter one allows to handle network management issues such as synchronization, network formation and maintenance (e.g. scan and association/disassociation), etc.
Generally, these services are exploited by the *Network Layer* to provide multi-hop through routing table or smart address managing in association/disassociation. There are a lot of Network Layers suitable for IEEE 802.15.4, with some examples reported in [9] and [43].
It is worth noting that by providing a Network Layer on the top of the IEEE 802.15.4 MAC Layer, we have just provided a software package that helps to hide the complexity and heterogeneity of the underlying hardware platforms and simplify the management of system resources: in other words, it could be considered as a minimal middleware. This remark is also compliant with the classification of middlewares for WSNs provided in [16] and [39]. Nevertheless, the middleware discussed in this paper is more complex with respect to the approach devised above. In addition to providing methods to route and control the medium access through IEEE 802.15.4, our framework embeds methods to ensure reliability of the network based on the exploitation of a hybrid cryptographic scheme and an intrusion detection system. It is worth to note that in [27], [28] and [29] we have proposed an architecture for a middleware where security services are embedded in the mobile agent-based middleware Agilla [1]. Here, part of the same considerations are moved in a different context for a different purpose: in [27], [28] and [29] the middleware is unaware of the underlying physical network, while in our current proposal security services are tailored to a system prototype that explicitly relies on the IEEE 802.15.4 standard.
This approach is quite different when compared to other related works. For example, *Zigbee* [43] aims to standardize the application execution environment covering the largest number of WSN application domains and providing some basic services. In fact, although Zigbee is famous for its definition of Network and Application Layers, to put on top of IEEE 802.15.4 ones, its specification includes a number of basic security provisions and options. In particular, Zigbee provides facilities to manage secure communications (for instance, link setup and key management), ciphering frames and controlling devices. Instead, the middleware proposed in this paper aims to provide advanced security services focusing on the most popular WSN application domain, i.e. monitor and control applications. In literature, other than ZigBee, there are several proposals of middleware platforms that provide security through cryptography: for instance, *SM-Sens* [14] uses symmetric and asymmetric cryptography along with message authentication code to ensure security requirements on data flows. It also provides a method to distribute keys by exploiting hierarchical routing and a mechanism to exclude compromised nodes of the network. *STaR* [10] is a modular, reconfigurable and transparent software component for secure communications in WSNs. STaR guarantees confidentiality, integrity, and authenticity by means of encryption and/or authentication and it is totally transparent to the application, i.e. no changes to the original application or the communication protocol are required. *SpartanRPC* [7] extends *nesC* programming language to provide a link-layer *Remote Procedure Call* (RPC) mechanism. All the RPC resources are protected

via language-level policy specification. *SMEPP Light* [37] features group management, group-level security policies, mechanisms for query injection and data collection based on a subscribe/event mechanism, and adaptable energy efficiency mechanisms. Other middleware proposals provide security by deploying specific defence mechanisms for a set of predefined attacks. *Di-Sec* [38], for example, provides a framework to model defence strategies. Through a training phase, nodes are able to learn the behaviour to adopt in case of attacks.

The middleware proposed in this paper exploits benefits of these two approaches (i.e. cryptography and attack defence mechanisms) by providing a light but powerful cryptographic scheme to protect data and an intrusion detection system to guarantee the availability of the network. Moreover, since it explicitly refers to IEEE 802.15.4-based WSN, the middleware is suitable for network composed by heterogeneous nodes. Usually, this property is not considered in middlewares that handle security but it turns out to become very interesting to deploy a real-world WSN.

3. Security in IEEE 802.15.4 networks

To discuss the security facilities provided by IEEE 802.15.4 it is important to highlight that the standard does not provide only a method to access the medium, but also several mechanisms to create and detect a WPAN, associate or disassociate with it and so on. This means that in a 802.15.4 network, there are several types of messages exchanged, such as *beacon packets* (used to synchronize the network), *acknowledgments packets* (used to notice the message reception), *data packets* and other *control packets*.

As specified by the standard [18], the MAC Layer provides security services on each incoming and outgoing frame (with the exception of acknowledgement packets). The services supported by the standard are as follows:

- *data confidentiality;*
- *data authenticity;*
- *replay protection.*

Data confidentiality is ensured by using encryption and decryption algorithms: the standard defines to use AES (*Advanced Encryption Standard*) with 128 bit keys. Data authenticity is guaranteed by using cryptographic hash functions that associate to each message a *Message Authentication Code* (MAC). The receiver can check it to authenticate the message. Finally, the usage of a monotonically increasing sequence number to each packet ensures the protection from replay attacks. It is worth noting that the standard can also work with no security, encryption only (AES-CTR), authentication only (AES-CBC-MAC), and encryption and authentication (AES-CCM). Each category that supports authentication comes in three variants depending on the size of the Message Authentication Code that it offers. Each variant is considered as a different security suite and has its own name. The Message Authentication Code can be either 4, 8, or 16 bytes long.

The IEEE 802.15.4 specification provides basic security mechanisms but these security features cannot work on their own: since the standard does not suggest any key management approach, in applications that require security a method to generate symmetric keys is needed. Symmetric key generation is one of the most addressed problems in the literature [5]. Pair-wise key pre-distribution solutions are based on deterministic pre-distribution of keys for each pair of nodes. Random pair-wise key schemes are based on storing only a subset of all possible keys in each node. To communicate with each other, each node needs to negotiate a key with its peer, randomly selecting one key in its subset [6]. The master key pre-distribution scheme requires that a master key is distributed in the entire network and that nodes use a combination of it and previous exchanged nonces [22]. Other schemes can be found in [5]. In [25], [29] and [30], we have proposed a family of novel schemes called TAKS (*Topology Authenticated Key Scheme*) to generate topology authenticated keys for handling cryptographic aspects in resource constrained deployments of WSN. TAKS cryptographic scheme allows to authenticate each message exchanged in the network referring to a certified topology of the network. Since TAKS provides good results from both performance and security points of view [25], its usage in IEEE 802.15.4 networks is very encouraged.

The complexity and distributed nature of a WSN makes cryptography not sufficient to ensure network security. In addition, to provide confidentiality, authenticity and integrity of messages, network security aims to make the system always (or mostly) available. Ensuring availability is more complex than other issues. Typically, this is done by auditing network activities, detecting potential threats and reacting opportunely through an Intrusion Detection System (IDS). IDS denotes a system that supports mechanisms to detect and appropriately manage (through reaction functions and proper countermeasures) intrusions and attacks in the form of malicious control and data messages [11]. An IDS is typically formed by three components: *Intrusion Detection* (ID) that deals with the detection of network intrusions by sensing suspect phenomena, *Intrusion Reaction Logic* (IRL) that schedules the priorities for actions on all compromised nodes according to a specific defensive strategy and *Intrusion Reaction Application* by performing the appropriate countermeasures (IRL Application).

In this paper, we focus on Intrusion Detection and we do not deal with Intrusion Reaction. For what concerns Intrusion Detection, an IDS can be classified into three frameworks: *anomaly based intrusion detection*, *misuse based intrusion detection* and *specification-based intrusion detection* [13], [19]. Anomaly based intrusion detection relies on the assumption that intruders will demonstrate abnormal behaviour relative to the legitimate nodes: anomaly has to be detected by knowing the normal

system behaviour. Instead, the misuse intrusion detection relies on the assumption of an up-to-date database of intrusion signatures. Using them, the system can easily detect intrusions on the network. Specification-based detection systems work by defining rules for attacks. Sensor node behaviour is checked against each rule sequentially. There is a failure counter associated with each node. If the sensor node violates a rule, a failure counter is incremented. If the number of failures of a particular node increases over a threshold after a time interval *t* an alert about that node is generated.
Another typical IDS classification is done with respect to the distribution of the code in the network [13]. There are the following types of IDS:

- in *purely distributed IDS* intrusion detection, algorithm is installed in every node;
- in *centralized IDS*, intrusion detection is performed only by the sink node or the base station upon the reception of processed information from the network;
- in *mixed distributed-centralized IDS*, suitable only for particular types of networks, such as clustered WSN, the detection is delegated to a particular subset of nodes of the network. Examples of these types of IDS are [21], [23] and [33].

Although many of these approaches can be applied in IEEE 802.15.4 WSNs, we cannot provide IDS examples focused on these networks: many works survey attacks and propose methods to detect them, such as [40], but, at the best of our knowledge, there are no papers proposing IDS frameworks specifically focused on these kinds of networks.

4. IEEE 802.15.4-based MW architecture

This section deals with the main functional blocks of the proposed IEEE 802.15.4-based middleware. A high-level representation is given in Figure 1.
The proposed architecture refers to a typical WSN protocol stack, where security facilities are now considered and embedded in the framework. From a protocol point of view, the proposed architecture specifies only MAC and Physical Layers to IEEE 802.15.4 and provides flexibility of choices on both Network and Application Layers. Usage of IEEE 802.15.4 is not limitative because it is the de-facto standard in industrial applications while Network Layer is not standardized as well, although Zigbee exploitation is quite diffuse.
In the following, the embedding of the security services into the reference middleware architecture is discussed. Typically, data security is accounted at the Application Layer: a large number of protocols, such as ones used on the *Internet*, provide security directly at Application Layer.

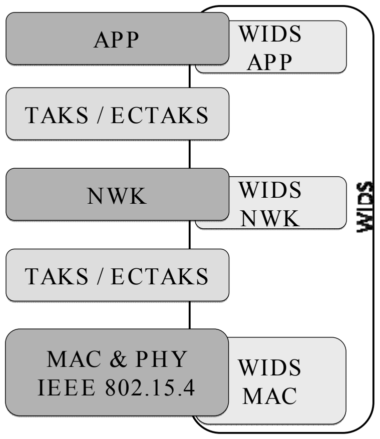


Figure 1. The middleware architecture

The proposed architecture is compliant to this view because it provides TAKS [30] facilities (ECTAKS if elliptic cryptography is exploited [25]) to Application Layer. From a Network Layer point of view, this means that only the effective payload (i.e. the application packet) is encrypted and only the intended legal receiver can decrypt it. However, IEEE 802.15.4 has some interesting properties that we can exploit to improve the security level. As we have seen in previous sections, 802.15.4 MAC Layer provides a security suite accessible by means of APIs provided by the MAC Layer itself. Using this service we can ensure the encryption of each MAC data packets (i.e. the entire MAC payload composed by Network and Application packets). We have designed the middleware so that encryption can be selected at one or both layers depending on user security requirements.
The proposed architecture motivates further remarks about IDS. Our consolidated line of research is oriented towards a *misuse-based purely distributed IDS* which exploits the *Weak Process Models* (WPM) over WSN, denoted here as WIDS (*WPM-based Intrusion Detection System*) [27], [28], [29]. First of all, WIDS is purely distributed. Most literature contributions propose to put intelligence (usually more consuming both in computational resources and in memory) outside the WSN [15], [42]: however, if the algorithms are designed by considering the very constrained environment of WSNs, these systems can operate as functionally "autonomous entities" and not only for pure sensing operations. This choice implies two types of benefits: the former one consists in the distributed architecture which avoids the typical drawbacks presented by centralized solutions; the latter one is the reduction in energy consumption since distributed solutions do not need sharing information with a centralized entity (i.e. sending them via radio and wasting energy). However, the drawback is that distributed IDS need a fine configuration.
Looking at the architecture, it is straightforward to remark the cross-layer nature of the Intrusion Detection System that concerns all active layers of the stack (i.e. Application, Network and MAC). Each active layer implements protocols characterized by a set of constraints

and rules and, for this reason, exposed to attacks by intruders: constraints and rules in a protocol represent points of weakness which can be exploited by intruders to induce altered behaviours on sensor nodes (e.g. a denial of service). For example, we can refer to a kind of attack known in literature as HELLO flooding. This kind of attack relies on the fact that wireless protocols often require that nodes execute an association procedure by sending the so-called HELLO messages. HELLO flooding is when the attacker continuously issues malformed HELLO messages to WSN nodes, which waste computational and memory resources that can later result in a denial of service. Now it is easy to understand how IDS is strictly based on the kind of protocol that it monitors. Therefore, the WIDS component of the middleware is conceptually the same one that we have proposed in [27], [28] and [29] but it is customized for the different protocols provided by the actual architecture. It is important to remember that, such an architecture, do not define any Network and Application Layer so, in this work, we do not provide any fixed approach to handle Intrusion Detection at this layer.

It is of paramount importance to remark that a specific solution needs to be tailored to each specific protocol stack. In this regard, we progress by assuming that MAC and physical layers are compliant to IEEE 802.15.4, while flexibility is assumed when considering network and application layers. Relying on the IEEE 802.15.4 is not to be intended as a restriction: instead, we claim the double perspective of i) devising, developing and testing a framework for a concrete case, and ii) impacting on the evolution of a de-facto standard in industrial applications and other domains. As for network layer and application layer only the following not restrictive requirements are assumed:

- *R1: the network layer shall not allow data transmission without association to a network;*
- *R2: the network layer shall implement a reactive routing protocol and it shall be aware of the association of new node to the network;*
- *R3: the application layer is concerned with the "monitoring and control" application domain, and it is supposed to resort on acknowledged transmissions.*

As a matter of fact, we can observe that several existing network layer routing protocols, such as [9] and [43], readily satisfy R1 and R2. At the same time, any monitoring and control application should be compliant to R3.
In the next two sections we will give a complete overview of both the cryptographic scheme TAKS and the WIDS design, focusing on the MAC Layer of the stack.

5. The cryptographic scheme TAKS

A high level description of the cryptographic scheme is firstly provided, along with an assessment of its robustness. Next, the implementation in the devised framework is presented and validated as well.

5.1. High level description

TAKS includes authentication, cryptographic key generation, and related management services. The authentication service is based on network topology in terms of neighbourhood relationships among parties. Each party can be authenticated if and only if the graph of the topological relationships between the related node and other nodes (not restricted to only one-hop neighbours) is included into the graph of authenticated topologies defined for that portion of the network. The function of *Certification Authority* (CA) is not centralized, but shared among all the parties in the network, in the sense that each party stores the minimum information to prove the authenticity of any signing party. Rationales and further details of the decentralized approach are provided in [24] and [30]. The topology-based mechanism also enables nomadic parties to be authenticated, as long as their local topological relationships are compliant to the authenticated ones: this approach allows to overcome typical limitations of purely position-based authentication mechanisms.
The service for cryptographic keys generation has been defined and is based on vector analysis over points in *Galois finite fields* with size of the order of the key length. It is a hybrid scheme that inherits some features of the *Diffie-Hellman* key exchange scheme [12]. It also inherits other features of the classical asymmetric paradigm public key-private key [17]. In summary, any secret share is generated in a party, starting from the partial key components stored in any authenticated pair. For very critical applications, security can be further enhanced by introducing special elliptical curves (specified in ANSI standards) as point generators over Galois finite fields. In this latter case a larger overhead is introduced.
While detailed descriptions can be found in [24] and [30], the basic behaviour is provided here:

- Each node stores a number of vectors over a predefined Galois finite field: these vectors define the graph of the planned authenticated topologies for that portion of a network (denoted with *Local Planned Topology*) and the partial key components, namely the *Local Key Component* and the *Transmitted Key Component*. The Local Key Component plays the role of the private key in an asymmetric scheme, while the Transmitted Key Component is intended as a public key. Once again, these vectors are not keys but only partial key components.

- The shared secret, i.e. the symmetric key, is computed from the key components associated to the parties involved in the data exchange. Key generation requires of course that a successful authentication of the parties has been completed. Once authenticated, the receiving party has gained the information to decrypt the received ciphered message by computing the shared secret from the Transmitted Key Component that the transmitting party has sent with the ciphered message and its own Local Key Component. The scheme is sketched in Figure 2 [24].

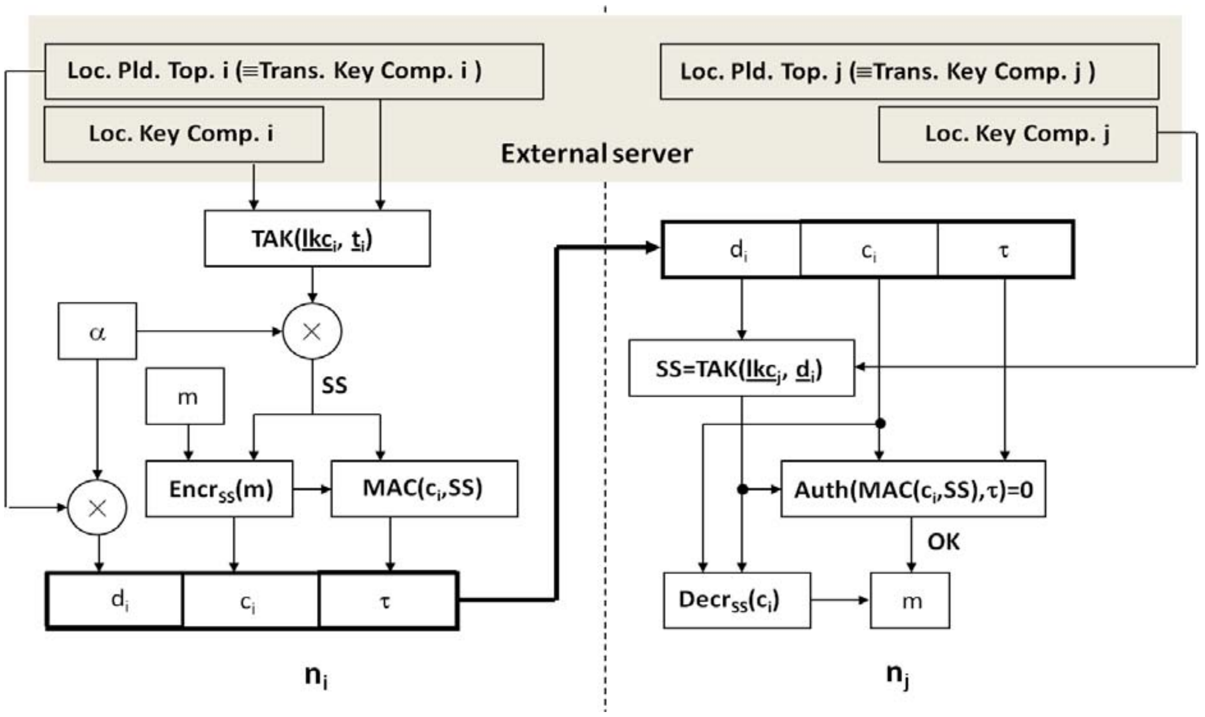


Figure 2. TAKS scheme

The setup protocol is 1-phase in the sense that the authentication phase and the generation phase are in fact executed in the same step, so that the transmitting party authentication information (i.e. the Local Planned Topology vector) is coincident with the key generation information (i.e. the Transmitted Key Component vector). This is different from the 2-phase scheme that we defined in an earlier work [30], where the authentication phase and the generation phase are separated. The *secret share* (SS) feeds both the coding and the MAC authentication algorithms to return the ciphered message and the authentication tag. The 1-phase transmission contains 3 information elements: the *ciphered message* (c), the *authentication tag* (τ) and the Transmitted Key Component (d). The receiving party computes its own version of the secret share SS' through its Local Key Component and d, then computes its own version of the authentication tag τ': in [32] it is shown that if and only if $\tau = \tau'$ then $SS = SS'$, hence the symmetric key is established between the parties and the message can be successfully deciphered by the destination party.

5.2. Security analysis

TAKS security analysis has been led by answering the following questions:

- Is TAK a real cryptographic key? I.e. which is the entropy per binit associated to TAK? A bit string can be regarded as a cryptographic key if its associated entropy per binit raises unity [34] so that each key behaves as a truly pseudo-random number. In [24] and [30] is shown that this requirement has been met in TAKS.
- How much a single node is secure, i.e. how much complex is the inverse problem to break TAK (security level in a single node)? In [30] has been proven that the security level in a single node is higher than the *discrete logarithm problem*: Equations show that the relationship between key components is not simply a discrete logarithm, which is one of most difficult problem in GF(q) algebra, because they appear both as multiplying factor of the exponentiation operator and in the exponent.
- How much a network is secure, i.e. how many nodes an attacker should compromise to break TAK (security level in the network)? This security level is calculated by evaluating the complexity to break the cryptographic key with all nodes in the network available. The T-Security concept is here introduced: given a network with N nodes, a cryptographic key is T-Secure if an attacker should capture $T+1<N$ nodes in the network to gain enough information to

crack the key. The best case is when T = N , because in this case the cryptographic key never can be violated as there is no enough information shared in the network to do that. This result can be achieved if a share of the information needed to generate cryptographic keys is external to the network (i.e. residing in an external server). In [30] has been analytically shown that TAKS is N-secure (i.e. T = N).

5.3. Design and implementation

The procedure including TAKS algorithms has been designed and implemented in terms of a set of modules on top of an intermediate bridge layer: this enables code re-usage across the implementations on top of network and MAC layers that are described in the following.

TAKS Layer

In order to enable easy integration with IEEE 802.15.4-based stacks, the TAKS modules have been designed by resorting to the typical paradigm of service primitives largely exploited also by IEEE standard [18]. As a consequence, any service exposed or required by TAKS can fall within one of the four generic types: request to require a service, indication to be informed about an internal layer event, response to complete a procedure previously invoked by an indication primitive, and confirm to convey the results of one or more associated previous service requests. Moving along this line, we were able to come up with the finite state machine in Figure 3 for the logic implemented on sensor nodes. In particular, for the specific interests of this section, we analyze how state machine evolves in packet transmission and reception.

It easily turns out from [24] that a packet transmission by means of a TAKS component requires more than one transmission across the wireless link: we call the sequence of packet exchanged by nodes as a *TAKS transaction*. Normally TAKS state machine is in IDLE state. In this condition, a state transition can only be induced by a TAKS transaction that starts. Of course, for a given node two different procedures are interested when a transmitting or a receiving transaction is considered.

When a transmission is required, TAKS component has to send both deciphering information and encrypted data. Therefore, the component first checks by means of the *lookup_destination()* function whether the transaction is allowed, i.e. it checks if it has the destination Topology Vector in his Local Planned Topology. If transmission is allowed, a random number is generated with *rand()* function and the deciphering information is calculated by multiplying the scalar random with the Topology Vector owned by the node. After calculating deciphering information, the component logic requests its transmission (event *a* in Figure 3) to the destination node of the original packet and it put the state machine in KT_TX_REQED state. The transmission result (event *b* in Figure 3) is notified by the corresponding confirm function. If an error is returned the previous procedure is rerun until to reach a threshold. If a successful transmission is notified, the corresponding TAK is computed by performing the *inner_product()* function on the Local Key Component and the destination Topology Vector, and then multiplying the resulting scalar with the opposite of the random scalar generated earlier. Such a scalar is then set as a ciphering key: this progression is maintained in the state machine evolving in the SET_KEY_REQED_OUT state. Also in this case, the correct key setting is notified by a confirm function (event *c* in Figure 3). The encrypted packet is transmitted only if the key has been successfully set (PKT_TX_REQED state). At the corresponding confirm reception (event *d* in Figure 3), the state PKT_TXED is achieved wherever the confirmation to higher layer is sent and the machine is automatically reset (event *e* in Figure 3).

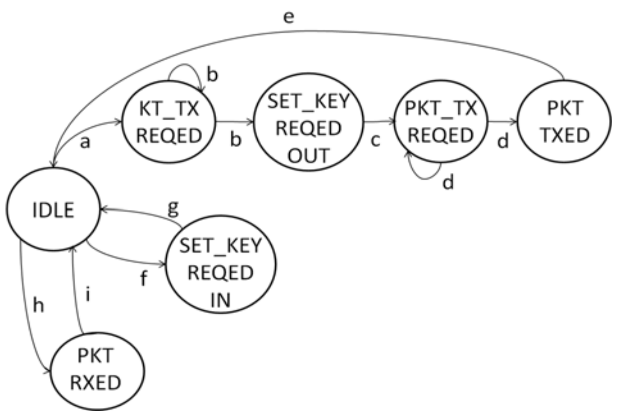


Figure 3. TAKS state machine implemented on the top of IEEE 802.15.4 MAC layer

When the reception is a considered in a node, the transaction is started by a plaintext packet reception in IDLE state (event *f* in Figure 3). When this packet is received by the TAKS component, the logic assumes that it contains deciphering information, so TAK is calculated as inner product between deciphering information and the Local Key Component owned by the node. This information is mapped on the state machine evolving in the SET_KEY_REQED_IN state. In this state the new calculated key setting is requested. At the occurrence of the corresponding confirm message (event *g* in Figure 3), the machine comes back to the IDLE state: the underlying layer is correctly set and it is only needed to wait for the encrypted packet. When the packet is received (event *h* in Figure 3), the decryption is automatically performed and, if the packet is also authenticated, it is also notified to the higher layer (state PKT_RXED). At this point, the machine is automatically reset in the IDLE state (event *i* in Figure 3).

The Bridge Layer

In this sub-section the main functionalities implemented by the bridge layer are discussed. Services provided by the bridge layer are transmission request and

corresponding confirmation, packet reception, and set key request and corresponding confirm. Implementation of these services is different depending on if they interface with MAC or network layer. While other services are rather standard and do not deserve specific descriptions, the set key request service on the top of MAC layer is of particular interest. In fact, IEEE 802.15.4 MAC layer handles security internally and interfacing with its primitives is needed. [18] defines an incoming and an outgoing frame security procedure and some parameters need to be set to allow a correct secured communications. In the following, a brief description is reported.
When a packet transmission is requested, MAC layer decides to secure the frame, i.e. to encrypt or authenticate it, according to a given security level. Moreover, a node stores a minimum security level for each other node with which it can communicate. When the MAC layer processes a received packet, it filters any of them with a security level score below a minimum security level. Since the TAKS procedure defines that deciphering information has to be transmitted in plaintext, we have set the minimum security levels to non restrictive values.
Another important parameter is the key identification mode. [18] defines how to store keys in a key table indexed by an identifier. This identifier can be explicit, if it is user defined, or implicit, if it can be implicitly deduced from addressing information. An explicit key identification implies a further overhead with respect to the normal security overhead due to identifier transmission. This enables the receiving side to know the identifier of the key to be used to decrypt the message. For this reason the implicit key mode is typically preferred and we have made this choice too in our implementation.

5.4. Cost and performance analysis

In this section, we report the cost analysis of the proposed scheme. This analysis aims to evaluate the complexity of the scheme regardless of encryption and decryption algorithms and message authentication coder (since they are always needed and could be freely selected by the network planner). As shown in [24], if *n* and *σ(i)* are the key size in bytes and the cardinality of the set of eligible neighbours respectively, then random generation and addition cost $O(n)$, multiplication and inner product cost $O(n^2)$ and the scan of all possible neighbours costs $O(\sigma(i))$.
Since a sending transaction is a serialized call of *rand()*, *lookup_destination()*, *inner_product()* and *multiply()*, we can affirm that the computational complexity of TAKS encryption is $O(n^2+\sigma(i)) \approx O(n^2)$. Similarly, since a receiving transaction is driven by the complexity of *inner_product()*, we can affirm that computational complexity of TAKS decryption is $O(n^2)$ as well. Moreover, the spatial complexity, due to the creation of temporary structures for mathematical calculations, is $O(n+\sigma(i))$.

The computational cost has been then evaluated in terms of execution time on the *Atmel Zigbit* [4] platform, that is provided of an 8 MHz clock (this is the reference platform for the WSN that has been deployed in our lab). With respect to the three main components involved by TAKS, i.e. deciphering information computation and key computation at both sender and receiver sides, the evaluated execution time turn out to be 5.1 ms, 7.2 ms and 4.8 ms, respectively. All these results have been obtained by using *IAR Workbench for AVR compiler* with medium level optimization.
To complete our analysis, we have measured the computation time for encryption, decryption and authentication tag generation time with the assumption that the cipher text length is one AES block: the times are 406 µs for both encryption and decryption, and 570 µs for authentication tag generation. It is worth noting that Zigbit radio component integrates an AES engine accessible via *Serial Peripheral Interface* (SPI) and performing hardware accelerated encryption and decryption [3]. For what concern energy consumption, we can observe that Zigbit microcontroller power consumption is negligible with respect to consumption of the transceiver in active modes [4]. This allows to approximate TAKS energy overhead to the TAKS radio overhead intended as the ratio between TAKS packet overhead and packet payload size of the typical packet exchanged in the network. Figure 4 shows relationship between TAKS overhead and packet payload size.
Supposing that the typical packet fits in two AES blocks the packet payload size is 32 bytes. If link-layer protection is activated, for each packet TAKS requires the transmission of the deciphering information (48 bytes). Moreover, if end-to-end encryption is activated additional 48 bytes are needed. Therefore, activating single facility, TAKS overhead (i.e. TAKS radio overhead) is 1.5x while activating both facilities at the same time the overhead is 3x. It is worth noting that above analysis refers to a case wherein each transmission requires a new key generation and then deciphering information needs to be sent at each packet transmission. Some caching techniques can be implemented to reduce overhead and energy consumption.

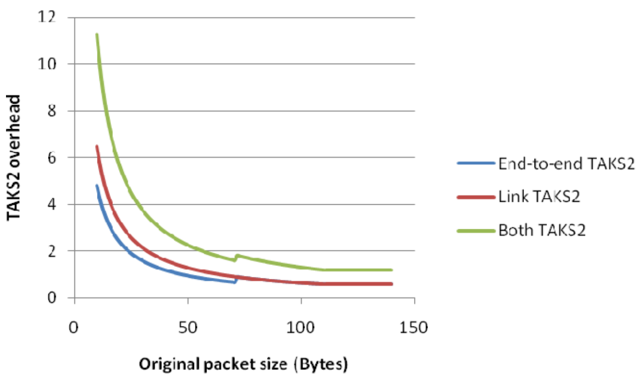


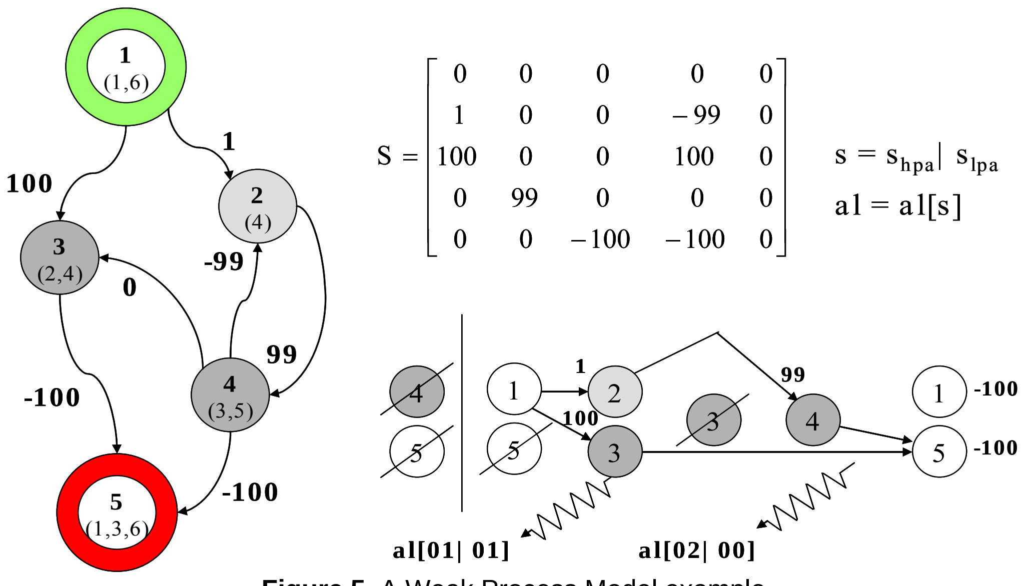
Figure 4. Relationship between TAKS overhead and packet payload size

6. WIDS-based intrusion detection

This section presents a description of our WPM-based IDS (WIDS), starting from a high level description and an analysis of security issues; then, a case study with details of our novel implementation is discussed. Finally, performance metrics are presented.

6.1 High level description

A *Weak Process Models* (WPM) is a particular sub-class of *Hidden Markov Chains*, where the many-to-many relationship between observables and states (which represent the possible phases during an attack engage) is made quite deterministic, by assigning to state transitions just 0/1 weights, rather than stochastic by assigning probability weights. This simplification avoids the needing of historical sequences of events, to derive the probability weights to assign to state transitions and, therefore, the implementation of a *Viterbi* estimators that is very expensive in terms of computational and memory requirements. Details on WIDS can be found in [32], here the basic behaviour will be provided. A WPM estimator computes the score associated to the set of possible state sequences corresponding to the sequence of observables that can occur during the time interval of interest. As shown in [32], the detection performance can be set arbitrarily high according to the accuracy of the threat representation. States in WPM are classified according to the estimated hazardousness of the threat in that state: *Low Potential Attack* (LPA) and *High Potential Attack* (HPA). We assume that until threats are confined in LPA states, no reactions should be issued. When in HPA, a reaction should be performed on the hazardousness weight (score) associated to the estimated state sequence. It is worth noting that, in line of principle, also unknown attacks can be detected whenever they show, during the time interval of interest, a behaviour in part similar to other classified attacks. To provide an outline of the WIDS behaviour, it is sufficient to consider WPMs as *Finite State Machines* (FSMs). During intrusion detection session, when predefined events occur, WIDS evaluates state transitions on WPMs modelling the database of attacks. When a HPA state is reached, the sequence of events is considered so serious to notify the IR component. An example of the overall process is sketched in Figure 5. The observables associated to each state in the WPM are represented into brackets. In Figure 5 are shown state transitions corresponding to the sequence of events (3,1,4,2,5,6). All the events not referable to initial state (green), final state (red) or to specific state transition compliant to the WPM are discarded. Alarms are raised when state 3 or 4 is reached.



$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -99 & 0 \\ 100 & 0 & 0 & 100 & 0 \\ 0 & 99 & 0 & 0 & 0 \\ 0 & 0 & -100 & -100 & 0 \end{bmatrix} \qquad \begin{aligned} s &= s_{hpa} | \, s_{lpa} \\ al &= al[s] \end{aligned}$$

Figure 5. A Weak Process Model example

6.2 Security analysis

WIDS security analysis has been approached by reducing ambiguous observables as much as possible. As formerly shown in [32], the rate for false positives (FPR) and false negatives (FNR) or mis-detections strongly depends on the accuracy of threat modelling by WPM. If any state sequence in WPM could be associated to a threat behaviour, FPR could be arbitrarily reduced. If any abnormal behaviour of the WSN could be mapped into a WPM state, FNR could be arbitrarily reduced too. Both the approaches comes at the cost of further states to be introduced, hence more computation and memory requirements. If these approaches cannot be fully adopted, alternative sub-optimal solutions can be considered as follow:

- Introducing further states associated to certain threat observables in paths where at least one state is associated to ambiguous observables. This approach surely lowers FPR and FNR, as the longer the path to detect hazardous events the more reliable would be an alarm. A drawback is that long paths, would reduce the reactivity in the monitoring service.

- Introducing a further class of states associated to ambiguous observables. This approach can reduce FPR or FNR as "ad-hoc" countermeasures can be applied to nodes where alarms from dubious observables are generated (e.g. node quarantine rather than link release).

6.3 Design and implementation

This section deals with the novel design and implementation of WIDS. Firstly, we define a realistic case study for WIDS application: from earlier assumptions, we derive a set of common attacks on which WIDS is built. Next, implementation issues as well as performance evaluation are detailed.

Case Study

Let us start with a survey of possible attacks that can be performed in a generic WSN. Then, we illustrate the case study defined to validate our IDS. Finally, we describe the aggregate WPM implemented in our IDS.
Following a misuse-based approach, the main issue is to define the set of threats to be modelled, as a good misuse-based IDS should necessarily have a good threats database. Then, it is possible to design and implement identified threats. Given the IDS layered architecture, threats can be classified into layers, from the physical layer to the application one. For what concerns physical layer, IEEE 802.15.4-based networks are subject to the typical wireless physical attack, i.e. *radio jamming*. [41] classifies jamming activities in *constant*, *deceptive*, *random* and *reactive jamming*. For what concerns IEEE 802.15.4 MAC layer, [35] and [41] report the following attacks: *link-layer jamming*, *backoff manipulation*, *replay-protection*, *nonce attack*, *GTS attack*, *ACK attack*.
The network layer is responsible of routing services, therefore a complete network layer attack list can be defined without referring to the specific protocol adopted by the layer itself. For example, [20] defines the following attack categories: *spoofed*, *altered*, or *replayed routing information*, *selective forwarding*, *sinkhole attack*, *sybil attack*, *wormhole*, *HELLO flood attack*.
Finally, since the WSN application layer is generally simpler, we neglect attacks at this layer; in any case those attacks should be considered as application protocol specific and are not considered here.
It is worth of mention that our case study covers the most important application context of WSNs, that is monitoring and control applications. When energy awareness cannot be neglected, IEEE 802.15.4 provides beacon-enabled networks, that are preferred to nonbeacon-enabled ones. To instantiate a beacon-enabled network to correctly handle energy consumption in every node, tree-based routing protocols have to be chosen. In such protocols, a reactive approach allows to route messages without wasting memory and without sending additional messages to populate routing tables.
Therefore, in our case study, only a subset of the attacks listed above can occur. For example, GTS attack is not considered, since it is too particular and not meaningful in the considered domain. Moreover, if TAKS is used, nonce attack cannot occur since keys are one shot. In fact, it should be noted that, in such a context, we require that a node performs the IEEE 802.15.4 MAC layer association to transmit data in the network. According to [18], the association procedure can be done by exploiting authentication features. If we plan to use MAC layer cryptography, attacks by means of spoofing, altering and replying routing information are not possible since messages are authenticated. Moreover, sinkhole and selective forwarding attacks can be considered as overlapping and most of the network layer attacks listed above can be actuated only if nodes are tampered and keys are stolen. However, this last situation can be detected by using our IDS with the aggregate threat model presented in Figure 6.

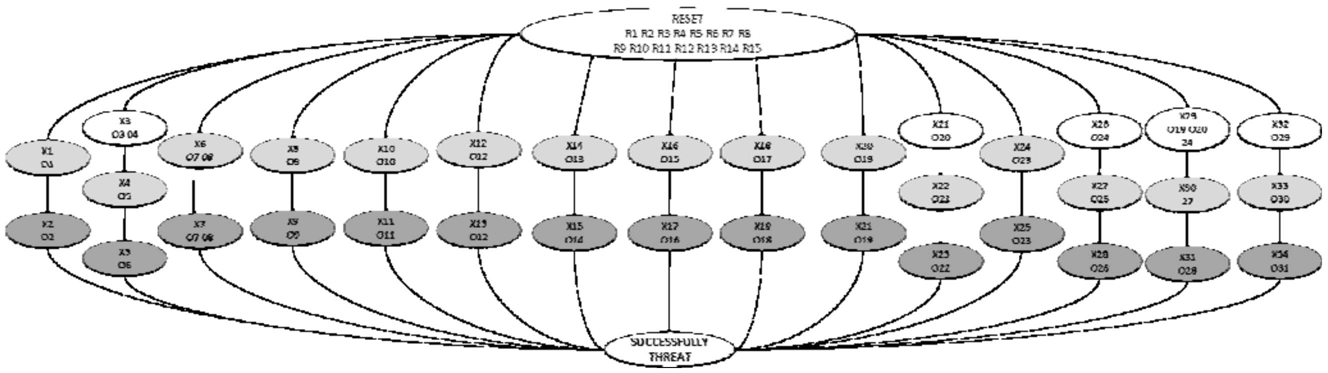


Figure 6. Aggregate WPM designed for threat detection

WIDS Layer

In this sub-section, the aggregate threat model presented in Figure 6 is described, by analyzing modelled threats and moving from the highest layer to the lowest one. It should be noted that our IDS currently allows to detect anomalies on control data only: some attacks based on node cloning and consequent admissible data injection are considered out of scope. In fact, this latter category of attacks can be detected by only analyzing data and using approaches such as the ones described in [31]. Therefore, in the present paper all the attacks based on correct but malicious data injection are excluded from our database. This also means that we can exclude application layer attacks since the two communication peers are always considered trusted. In the following, network layer attacks are firstly considered.

Selective forwarding. Routing algorithms are based on the assumption that the nodes forward received messages. In a selective forwarding attack, malicious nodes do not forward certain messages and ensure that they are not propagated further. Many approaches are proposed in literature to detect selective forwarding attacks. As in [32], we exploit the TAKS authenticated acknowledgment packet sent by the application layer to check if it is compliant to R3. Deterministic selective forwarding attacks can be easily detected with retransmission, while random selective forwarding needs statistic analysis. In Figure 6, the observable O1 means that an application layer acknowledgment has not been received and so a LPA is raised. If the number of this anomalies exceeds a programmable threshold (observable O2), an HPA is raised. Packet transmissions and receptions are mapped with observable O3 and O4 respectively and, if their difference exceeds a threshold (observable O5), an LPA is raised. If this occurs several times (observable O6) the HPA is raised.

Sinkhole. In a sinkhole attack, the adversary goal is to lure nearly all the traffic from a particular area through a compromised node. With a tree-based reactive routing protocol, sinkhole is equivalent to selective forwarding wherein all packets are dropped, therefore it is already manageable with previous countermeasures.

Sybil. In this attack, a single node presents multiple identities to other nodes in the network. According with requirement R1, a node can communicate only if it has been associated to a coordinator, using TAKS procedures to perform an authenticated association. To associate with a coordinator, a device has to send his ephemeral *Topology Vector* (i.e. the deciphering information) whose direction (i.e. vector direction) is unique in a network. We should distinguish two cases: the case where the attacker uses the same topology vector for its different identities and the case where the attacker uses different topology vectors for its different identities. Both cases can be easily detected. In the former one, the attack can be detected by maintaining associated information for the *Local Planned Topology* in each coordinator. By checking this information when an association request is received, it is possible to detect the attack. In the latter case, an attacker cannot use neighbour topology vector because the attack can be detected as in the former case. For this reason, attacker should use a topology vector not compliant to the planned topology and so the coordinator cannot build a correct key and authenticate the received authentication request. Both anomalies are represented in Figure 6 with observables O7 and O8 respectively. If they occur, a LPA is raised and if they occur again a HPA is launched.

Wormhole. In the wormhole attack, an adversary tunnels messages received in one part of the network and replies them in a different part. Then, this attack can be performed only if at least two nodes have been totally compromised. Assuming wormhole attack between two non overlapping sub-trees as unique meaningful case, the attack can be easily detected. Our routing protocol implies a unique routing path; furthermore, an application layer address uniquely defines a network layer one. Therefore, a wormhole attack can be always detected by checking the correctness of the sub-tree from which a message is received, i.e. checking if the node from which the message is received can be a router for the node identified by the application layer address stored in the message itself. In Figure 6, detection of previous anomaly is represented by the observable O9. If it occurs, immediately a LPA is raised and, if it occurs again, a HPA is launched.

Hello flood. In Hello flood attack, the adversary convinces network nodes that it can provide a very high-quality route to the base station but those nodes sufficiently far away from the adversary would be sending packets into oblivion. Several approaches to detect hello flood aim to check the link bi-directionality, e.g. [32]. Differently from some other protocols, IEEE 802.15.4 MAC layer association procedure requires a bi-directional exchange of packets and the association can be completed only if the link is stable. However, when the link is not stable, the association procedure fails in a certain timeout. Usually, this event (represented in Figure 6 with observable O10) can be considered as an anomaly and a LPA is launched when it happens. If this anomaly repeats for a certain number of times (observable O11), a HPA is raised.

MAC layer attacks are considered in the following.

ACK Attack. Non broadcast transmissions between two IEEE 802.15.4 MAC peers are generally characterized by an acknowledgment by the receiver. Acknowledgment packets are not encrypted and can be linked to the transmitted packet by means of a sequence number. An eavesdropper can listen to the un-encrypted sequence number, corrupt the frame by interferencing at receiving time and send a fake ACK frame with the related sequence number to the sender in order to fool the sender as if the ACK were coming from the receiver. Therefore, a sender cannot be sure if the received frame is either coming from the intended receiver or from another node. Since there is no integrity protection provided on ACK frames, this weakness should be addressed at higher layers, e.g. by making the receiver able to send back an authenticated acknowledgement to the sender. In our case study, an acknowledgment attack can be performed in two cases: on a command frame (specifically on association frame) or on a data frame. MAC commands are generally used to request information to another peer (e.g.

association response, data request, etc.) so they initiate procedures that use timeouts to be fault-tolerant. Elapsing of these timeouts should be monitored to detect anomalies. In the data frame case, although it is possible to provide a link-layer authenticated acknowledgement, we have chosen to implement only the end-to-end alternative, since it allows to save energy and to detect anyway this anomaly even if less reactively. For these reasons, we do not need further monitoring to detect ACK attacks.

Replay-protection attack. Replay protection mechanism is a built-in mechanism provided by IEEE 802.15.4 that allows to detect the replay attack, i.e. the retransmission of a previously transmitted packet. It is implemented by sending a frame counter together with the packet. At each packet reception, the frame counter is compared with the previously obtained counter. If the current counter is equal to or less than the previous one, then the frame is rejected. However, the replay protection mechanism provided in the IEEE 802.15.4 specification is subjected to replay-protection attack which can be accomplished by an adversary by sending many frames containing large counters to a legitimate receiver. When another legitimate sender transmits a frame with a lower counter, it will be rejected according to replay protection procedure. Detection of this attack is straightforward: it is sufficient to monitor frame counters and raise an alarm when anomalies are detected. This can be observed also by WPM of Figure 6: if a frame counter anomaly (i.e. observable O12) is detected a LPA alarm is raised, and if it occurs again a HPA is delivered.

Backoff manipulation. This attack is realized by means of a node that transmits valid packets cheating applying backoff algorithm. Given a specific network, it is possible to simulate the network behaviour using proper IEEE 802.15.4 models, such as [8] in OMNET. In particular, from these simulations, we can estimate parameters such as the maximum number of attempts for transmission (let this number be *n*). Therefore, a backoff manipulation attack can be detected by monitoring the number of attempts that a node performs before to transmit. In our case study, we have planned to monitor transmissions that require *n-2*, *n-1* and more in excess of *n* tentatives. All this three events are considered as anomalies; when they happen, an LPA is raised (observables O13, O15 and O17 respectively) and, if the number of attempts exceeds a programmable threshold (observable O14, O16 and O18 respectively), a HPA is delivered.

Link-layer jamming. Link layer jamming is a more complicated type of attack among the jamming ones. An intelligent adversary, who wisely uses the link layer protocol logics, can be as defective as a blind radio jammer but consuming less energy and decreasing probability to be detected. For example, in IEEE 802.15.4 networks it is possible to be reactive to beacon transmission or other frame transmission, i.e. command or data frames. Both cases can be easily detected: in the former case by using the same approach used for random jamming and in the latter case by monitoring data transitions that fail. In Figure 6, observable O29 means link layer acknowledgement packet lost and, if the number of lost packets exceeds a threshold, a LPA is raised. If this occurs again the corresponding HPA is delivered.

In the following, we consider attacks at the physical layer (i.e. jamming). [41] classifies this type of attack in constant jamming, deceptive jamming, reactive jamming, random jamming and link-layer jamming.

Constant jamming. The constant jammer continuously emits a radio signal. For this reason, it can be detected only in transmission mode, since it does not allow any transmission and then reception in radio range area. IEEE 802.15.4 MAC layer Specification [18] defines that some packets (i.e. *beacon* and *ACK* packet) do not require CSMA/CA for their transmission. In fact, specific timing constraints ensure the reception of these packets. Therefore, continuous jamming can be detected by performing the clear channel assessment before sending beacon or ACK packet. Nevertheless, some strict timing constraints on ACK packet transmission imply to exclude them from our strategy and then, only clear channel assessment before sending beacon is exploited. Moreover, a constant jamming implies that transmission *Quality of Service* (QoS) is surely not satisfied. This means that it can be also recognized by monitoring the transmission rate: if it is under a certain threshold, an anomaly can be detected. Summarizing, if a clear channel assessment before beacon transmission is unsuccessful (observable O19), a LPA is raised and, if it occurs again in a certain timeout, a HPA is launched. Moreover, constant jamming can be detected if the number of clear channel assessment failures (observable O20) exceeds a defined threshold (observable O21). When this occurs, a LPA is raised and if it occurs again (observable 022), a HPA is delivered.

Deceptive jamming. This attack means that a node continuously injects regular packets to the channel without any gap between subsequent packet transmissions. It should be noted that infinite preamble transmission is considered as a deceptive jamming, so a distinction is needed between the case when a packet jamming is passed by physical layer to an upper layer or not. In the former case, a deceptive jamming causes the reception data queue saturation while the latter case can be handled as a constant jamming. For this reason, deceptive jamming can be detected by adding to the policy for constant jamming also the monitoring of received message queue saturation (observable O23). If it is observed, a LPA is raised and if it occurs again a HPA is launched.

Reactive jamming. The reactive jammer stays quiet when the channel is idle, but starts transmitting a radio signal as soon as it senses activity on the channel. For this reason, reactive jamming can be detected only by the node that receives corrupted packets. By analyzing timing distribution of corrupted packets, reactive jamming can be easily detected in turn. In particular, in Fig. 3 the observable O24 means that a corrupted packet has been received. It is worth noting that this observable cannot be considered as an anomaly, so alarm is not raised. But if a threshold is exceeded (that is represented by observable O25), a LPA is delivered. If this happens a number of times that exceeds another programmable threshold (observable O26), a HPA is raised.

Random jamming. Random jamming takes place when the attacker transmits and sleeps randomly. Given its feature, random jamming can be detected by using a proper mix of the previous approaches: clear channel assessment failed threshold, received corrupted packets threshold and beacon corruption can be evaluated together and the alarm raised if a weighted sum of these events exceeds another threshold. This approach is implemented with a WPM identified by X29, X30 and X31 states.

Implementation issues

From a user perspective the ID component discussed in this paper can be seen as a black box that is able to provide both LPA and HPA notifications. To accomplish this service, the component only needs to know if a certain event occurs. Therefore, the APIs provided by the component are *input_occured()*, *raise_LPA()* and *raise_HPA()*. While events of interest have been surveyed in the previous sub-section for each kind of attack, design and implementation issues are discussed in this sub-section.
As already stated, WPMs are particularly appealing from an implementation perspective: in particular, conventional implementation techniques for state machines can be still applied. A finite state machine can be implemented directly in the code or by using proper data structures. The latter choice gives some more pros with respect to the former one, as we pointed out in [24].
Therefore, we defined the data structure shown in Figure 7 to represent a WPM with a state space composed of 4 states, an observable space composed of 3 observables and the following state-observable relationship.

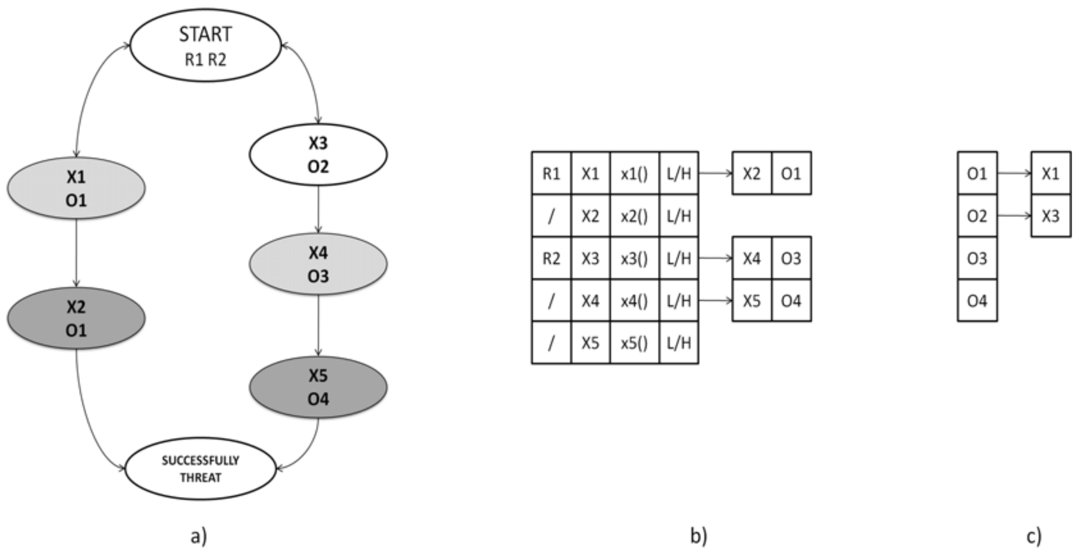


Figure 7. A WPM example and its corresponding implementation

The data structure stores for each state Xi:

- if it is a LPA, a HPA or none;
- a handler to execute a proper function when Xi is reached;
- its adjacency list, i.e. the list of states reachable from Xi and the observable that allows state change.

Figure 7 also includes a graphical sketch of the WPM implementation model. We can observe that state X2 or X4 is reachable from state X1 or X3 if observable O1 or O3 is revealed. Therefore, the first and third lists contain the couples (X2, O1) and (X4, O3).
In the following, a WPM is modelled as a dynamic process where observables act as inputs to the process itself. By exploiting the data structure defined before, the computation of the system dynamics is performed as follows: the current state is stored in a proper variable (its default value is reset state X0) and, when an observable is revealed, say Oi, the list associated to the current state is scanned while checking for the item (Xi, Oi). If it is found, the current state is updated to Xi. This approach allows to shorten the computation time with respect to the state space representation approach.
As hidden state machines, more states can be associated to a single observable and therefore, given an observable sequence on a WPM, more state traces can co-exist. Thus, it is important to provide a container structure to save multiple traces. We have adopted a bidirectional list to obtain good performance in both trace inserting and deleting.

How to save each trace in memory is another key point. By exploiting Markov property (i.e. each attack is modelled as a memory-less process), we can save only the actual state for each trace. Then, in the following we denote with *lsi* the last state of trace *i* maintained in the bidirectional list. This choice has an interesting consequence: since current states of current traces are maintained in memory to allow the algorithm to evolve, the score can be considered unnecessary. In fact, score calculus (that could result expensive since it requires arithmetic operations), can be replaced by a lighter state typology checking.
Therefore, when an observable Oi occurs, i.e. when *input_occurred()* is called:

- all stored traces are tested, i.e. for i from 0 to size of the stored traces set, if there is a couple (Xi, Oi) in the adjacency list associated to the last state of the trace lsi, the last state of the trace *lsi* is updated to Xi. Xi state type is checked and an alarm is raised if it is a LPA or HPA state calling raise_LPA() or raise_HPA();
- a new trace is tested, i.e. if there is a couple (X, O) in the adjacency list of initial state X0, a new trace is created and inserted in the set. In both cases, eventual handler associated to x is called.

This approach is very suitable for states with a very short adjacency list: when this condition is satisfied, the procedure results very efficient. When the adjacency list is not too short, the scan procedure to find new states could be long and thus too expensive. In this situation, exploiting an indexing data structure for observable instead of state as in Figure 7 could be more convenient. This consideration is exploited to store adjacency list of *reset state*: as showed by the WPM of Figure 6, about half of any WPM states are linked to the reset state and this implementation strategy yields both memory saving and better computation performance. Moreover, since most of the WPM states have associated a reset timeout, we have chosen to store for each state an additional information representing possible reset observable. Intuitively, this allows to save a lot of memory by observing that WPMs are generally not deep trees and that initial states have generally a reset observable associated to them.

6.4 Cost and performance analysis

In this sub-section, we describe the cost analysis of the proposed ID component and its execution time. This means that *input_occurred()* computational complexity and spatial complexity of the data structure representing WPM discussed in the previous sub-section are both discussed.
For what concerns *input_occurred()* cost, two phases need to be distinguished: a former phase that aims to update existing traces and a latter one that evaluates new traces creation by analyzing the reset state adjacency.
Current trace updating consists in a trace scanning to check if the occurred observable determines a trace update. For each stored trace, adjacency list of corresponding last state is checked: therefore, if the worst case is considered, this task costs *O(t*n)*, where *t* is the maximum number of traces and *n* the number of WPM states. Nevertheless, in a real WPM state adjacency list size can be considered constant with respect to the number of states, so in real cases computational complexity can be approximated to *O(t)*.
Instead, new trace creation consists in a scanning of reset state adjacency list. We remember that, for reset state, differently from other ones, adjacency list is an array where each position corresponds to an observable; furthermore, it points to the state list reachable from the reset state if corresponding observable occurs. With this data structure, the second phase cost in worst case assumption is *O(n)*. Nevertheless, the cost can be considered linear only when WPM moves from reset state with a limited number of observables. In real cases, this is not verified and the size of lists associated to observables can be considered constant. Therefore, in meaningful cases, second phase computational complexity can be approximated to *O(1)*.
Finally, *input_occurred()* computational complexity is *O(t*n)* in the worst case, but can be considered equal to *O(t)* in real-world application.
For what concerns spatial complexity, the following expression approximates memory occupancy (expressed in byte) with *n* equals to states number, *ni* equals to initial states, *o* represents number of observables and *t* is the maximum traces number:

$$S(n, ni, o, t) = 7(n-1) + 4(n-ni) + 2o + 3ni + 10t$$

Therefore, the spatial complexity of our ID component is *S(n+o)*, since *n* and *o* upper bound both *ni* and *t*. Applying the above formula to WPM represented in Figure 6 and assuming *t* equals to 5, the total data memory required by the component is 448 bytes.
Our IDS performances have been evaluated on *Atmel Zigbit* [4] nodes running with 8 MHz clock, considering the two phases of the algorithm, i.e. traces updating and creation.
For what concerns trace updating function, given a generic input and a trace, it is necessary to distinguish two cases: an update hit when the input implies a real update of the trace and an update miss when function simply verifies that input does not imply a real update. In the first case, the function spends near 85 μs to accomplish its task, while in the second case only about 17 μs are needed. This figure has been obtained under the assumption of only one trace lying in memory. Therefore with the assumption of five traces in memory, we have verified that the entire task requires about 150 μs, since it performs four update miss and an update hit.
For what concerns trace creation function, the same consideration holds true: also in this case, an input can imply a trace creation hit or miss. When the observable (i.e. the input to the function) requires a single trace creation, the function spends 100 μs approximately to

accomplish its task, while it spends about 25 µs in trace creation miss case. Moreover, if trace creation requires a trace deletion due to trace number threshold exceeding, additional 30 µs are needed.
Overall, for the WPM considered, the worst case is when double trace creation is required with trace memory saturated: in this case five update miss occurs followed by two traces creation with deletion. In this case, the IDS algorithm is performed in only 350 µs.

7. Validation

This section deals with validation of the implementations of the cryptographic scheme TAKS and the intrusion detection system WIDS described in previous sections.
Main TAKS functionalities have been already validated in [24] and [30] where a TAKS implementation for *TinyOS* [36] has been proposed. With respect to [24], the scheme discussed in this work is characterized by two different phases: a key agreement with the transmission of deciphering information and the effective data transmission. We have validated the novel state machine by checking the correctness of all possible paths. In particular, two aspects have been taken into account.
The first issue regards fault tolerance with respect to packet transmission and reception errors. For example, it has been checked that errors such as transmissions to not active nodes, do not compromise the subsequent communications: indeed, although in Figure 3 this aspect is omitted to simplify the discussion, when packets retransmissions exceed a programmable threshold, the state machine is reset and able to accept new communications. Failures due to this kind of transmission error are very simple to reproduce and then correct behaviour verification is straightforward: we have verified that nodes sending data are able to start several TAKS transactions even if they have not been correctly completed. Since packet retransmission is also adopted for channel access failures, this verification has allowed to verify (by design) TAKS with respect to this kind of error. Instead, when we consider reception errors we need to verify that the state machine is able to receive packets following an abnormally completed TAKS transaction. Since we cannot have a data transmission without a key agreement between parties, we need to verify that nodes are able to perform two consecutive agreements. This behaviour has been verified by using two nodes (i.e. a transmitter and a receiver) and by forcing the transmitter to send two different key agreement packets for each transaction. In this condition, the receiver has been able to correctly decrypt each message exchanged with the transmitter.
Previous tests allow to verify correctness of TAKS implementation in very simple networks composed by only two nodes. But when network grows, we have more transmissions and receptions occurring with simultaneous transactions. In each single node, transmission operations are serialized and then concurrent outgoing transactions are not possible. Instead, transactions interleaving could happen at the receiver side since it is possible that two nodes start TAKS operations simultaneously towards the same receiver. This aspect has been checked by means of a stress test. A test network has been setup, which is composed by a receiver acting as a 802.15.4 coordinator, two transmitters acting as devices, and a sniffer. Two transmitting nodes have been configured to continuously send known and progressive strings, and the receiver has been configured to send decrypted strings on its serial port. When analyzing the messages exchanged by nodes logged by the sniffer node, we have been able to check that our TAKS implementation properly handles packet interleaving. We have observed that the coordinator has been able to correctly decrypt messages even if TAKS transactions were interleaved. Given the result of the previous tests, we can consider TAKS suitable for arbitrary size networks.
For what concerns WIDS, validation has concerned two different aspects: the former one is related to check correctness of the implemented algorithm, the latter one is related to the validation on real nodes soliciting IDS with real attackers.
To check IDS correctness we have focused on its underlying algorithm. We have tested on a personal computer (PC) exactly the same implementation of the algorithm running on the Zigbit platform: this has been possible by simulating the behaviour of Atmel stack on a PC while replacing the stack itself with a software library offering the same *Application Programming Interface* (API) offered by the Atmel component. Fortunately, the IDS algorithm exploits only few of the operating system services provided by the stack, i.e. timers to reset traces. In fact, by operating with the Atmel stack, the operating system services are similar to services primitives described in section 5.3.1, i.e. responses to a request are provided as deferred or asynchronous callbacks. To provide an event driven programming approach exploiting the PC multithreading environment is straightforward: each request creates a new thread that introduces a specific delay and then calls the proper callback. By exploiting this environment, we are able to simulate the behaviour of the Atmel stack with the availability of all useful operating system services (e.g. terminal, accurate timers, file system, etc.). As an example, Figure 8 shows the code of API functions implemented to simulate timer behaviour.
Several tests have been conducted to study the evolution of the algorithm. Each of them has required: 1) the design of a specific sequence of observables that are suitable to solicit part of the algorithm; 2) the coding and consequent running of such sequences in the simulated environment.
It is worth noting that, in the simulated environment, each test appears as a series of calls to *input_occurred()* and to the sleep function offered by the PC operating system. They are then simpler to code with respect to direct operations on Atmel stack. As discussed in Section 6.4, a complete coverage of the algorithm can be obtained by analyzing trace creation and updating.

```
/*Thread's code*/
void* wait_for_ms(void* arg)
{
  mytimer_t p;
  p.id = ((mytimer_t*)arg)->id;
  p.handler = ((mytimer_t*)arg)->handler;
  p.timeout = ((mytimer_t*)arg)->timeout;
  free(arg);
  struct timespec req, rem;

  rem.tv_sec = p.timeout / 1000000;
  rem.tv_nsec = p.timeout % 1000000;

  do
  {
    memcpy(&req,&rem,sizeof(struct timespec));
    nanosleep(&req, &rem);
  }while((rem.tv_sec !=0) && (rem.tv_nsec != 0));

  p.handler();

  pthread_mutex_lock(&mutex);
  timers[p.id] = 0;
  pthread_mutex_unlock(&mutex);
  pthread_exit(NULL);
  return NULL;
}

/*Simulated start timer API command*/
bool bridge_layer_start_timer(uint8_t id, uint32_t timeout,
     void (* handler) (void))
{
  if((id < MAX_TIMERS) && (id != 0))
  {
    mytimer_t* p = malloc(sizeof(mytimer_t));
    p->id = id;
    p->handler = handler;
    p->timeout = timeout;

    pthread_mutex_lock(&mutex);
    pthread_create(&timers[id], NULL, wait_for_ms, p);
    pthread_mutex_unlock(&mutex);
    return true;
  }
  return false;
}
```

Figure 8. API functions to simulate timer behaviour

Therefore, trace creation hit or miss and trace update hit or miss have been solicited in the simulated environment and their correctness has been checked. The detection of all the attacks has been validated at simulation level providing ad-hoc and random sequences of observables in input to the algorithm and analyzing its responses.
This first kind of validation has allowed to functionally validate the algorithm, but it does not allow to completely validate IDS since it should be running on nodes. For this reason, a second phase validation has been conducted directly on nodes and soliciting them with some real attacks.
For example, the radio component mounted on board of a Zigbit platform provides *Continuous Transmission Test Mode* that we have exploited to validate several attacks based on jamming. Trivially, it has been possible to build a continuous jammer node by simply activating this mode. Then, to validate detection of continuous jamming attack, we have built a network composed by two nodes, specifically a device exchanging data with a coordinator and both them running WIDS. When we have turned on the jammer in the range of the nodes, both of them have detected the attack although in different ways: the coordinator assessing the channel before transmitting beacons and the device by means of the threshold on retransmissions.
Instead, by interleaving Continuous Transmission Test Mode with idle mode according to random generated time periods, it has been possible to realize a random jammer. Random jamming attack detection has been tested in the same conditions described above and with the same result. Both nodes have detected the attack adopting the same strategies even if detection of this attack requires more time than continuous jamming depending on the pseudorandom sequence generated by the attacker.
Instead, by enabling the transmission mode for a certain time at the reception of any packet, we have validated reactive jamming. This attack has mainly affected acknowledgement receptions since these packets are transmitted without access control. In our test network, only the device node has been able to detect this attack since it is the only node that requires acknowledgement reception.
When dealing with MAC layer attacks, Continuous Transmission Test Mode has been used again to validate backoff manipulation attack by enabling transmission at each beacon reception for a certain time. In our test network and in the absence of an attack, we can assume that a device node does not assess the channel more than two times for each transmission. In fact, [18] sets to 2 the number of backoff periods that need to be clear of channel activity before the transmission can start. By monitoring the transmissions that require more than two backoff periods, we have been able to efficiently detect the backoff manipulation attack as explained in Section 6.3.2. Obviously, this parameter should be properly set when network size increases.
Another MAC attack that we have decided to reproduce is the replay attacker: it is very simple to reproduce since Atmel MAC layer handles frame counter via software and then the attacker can be obtained by deleting the instruction responsible of this increment. In this manner, the attacker node sends packets with the same frame counter, thus implying the occurrence of the observable O12 on a node running WIDS. We have verified that, if we program the attacker node to send messages to one of the two nodes in the test network, the interested node is able to detect the attack.

8. Conclusions and future works

The middleware proposed in this paper aims to provide advanced security services to applications that rely on a real-world WSN protocol stack where lower layers are compliant to the IEEE 802.15.4 standard. Our approach makes this middleware suitable for a wide set of applications since IEEE 802.15.4 is the de-facto standard used for realistic WSN deployment. Our work proposes TAKS adaption to be implemented on the top of the IEEE 802.15.4 MAC Layer (i.e. topology-based key agreement protocol for 802.15.4 networks) and a set of design choices that we have identified to implement WIDS while guaranteeing availability of the network.
Performances have also been measured. It has been shown that, on the ZigBit microcontroller (i.e. *Atmel ATmega1281* running at 8 MHz), TAKS execution time is

about 10 ms. TAKS introduces a radio overhead factor of 3x or 1.5x depending on whether both end-to-end and link-layer cryptography are activated simultaneously or not. On the same microcontroller WIDS evolves in only 350 µs and requires only 448 bytes memory space. The proposed middleware has been implemented and its validation has been carried out by emulating external stimulus and collecting the responses of various algorithmic components in elementary scenarios. Nevertheless, an extensive validation has been planned in a large scale test-bed that we are deploying in our departmental facilities.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreements n° 257462 HYCON2 Network of excellence and n° 240555 ERC SG VISION. The development of the middleware platform also fits in the frame of "Ricostruire" and "SMILING" projects supported by the Italian Ministry of Economic Development to enhance technology transfer in the RIDITT framework.

References

[1] AGILLA Home Page, http: // mobilab.wustl.edu / projects / agilla /, 2015.
[2] I. F. Akyildiz IF, W. Su,Y. Sankarasubramaniam, E. Cayirci, "A survey on sensor networks", IEEE Communications Magazine, 2002.
[3] Atmel Corporation, "AT86RF212 Data Sheet", www.atmel.com / images / doc8168.pdf, 2015.
[4] Atmel Corporation, "ZigBit™ 700/800/900 MHz Wireless Modules - ATZB-900-B0 Datasheet", www.atmel.com / Images / doc8227.pdf, 2015.
[5] S. A. Camtepe, B. Yener, "Key Distribution Mechanisms for Wireless Sensor Networks: A Survey", Techical Report TR-05-07, 2005.
[6] H. Chan, A. Perrig, D. Song, "Random Key Predistribution Schemes for Sensor Networks," IEEE Symposium on Research in Security and Privacy, 2003.
[7] P. Chapin, C. Skalka, "SpartanRPC: Secure WSN middleware for cooperating domains", Proceedings of the Seventh IEEE International Conference on Mobile Ad-hoc and Sensor Systems, 2010.
[8] F. Chen, F. Dressler, "A Simulation Model of IEEE 802.15.4 in OMNeT++," Proc. of FGSN - Poster Session, Jul. 2007.
[9] A. Cunha, M. Alves, A. Koubaa, "Implementation Details of the Time Division Frame Scheduling Approach for Zigbee Cluster-Tree Networks", IPP-HURRAY Technical Report, HURRAY-TR-070102, Jan 2007.
[10] R. Daidone, G. Dini, M. Tiloca, "STaR: a Reconfigurable and Transparent middleware for WSNs security", Proceedings of the 2nd International Conference on Sensor Networks (SENSORNETS 2013), 2013.
[11] D. E. Denning, "An Intrusion-Detection Model", IEEE Transactions on Software Engineering, vol. SE-13, no. 2, Feb. 1987.
[12] W. Diffie, M. E. Hellman, "New Directions in Cryptography," IEEE Trans. Information Theory, vol. 22, no. 6, pp. 644-654, Nov. 1976.
[13] A. H. Farooqi, F. A. Khan, "Intrusion Detection Systems for Wireless Sensor Networks: A Survey", Communication and Networking, pp. 234-241, 2009.
[14] L.H. Freitas, K.A. Bispo, N.S. Rosa, P.R.F. Cunha, "SM-Sens: Security middleware for Wireless Sensor Networks", Proceedings of the Information Infrastructure Symposium, 2009.
[15] S. Gupta, R. Zheng, A, M.K. Cheng, "An Anomaly Detection System for Wireless Sensor Networks", Proceedings of IEEE International Conference on Mobile Ad hoc and Sensor Systems, 2007.
[16] S. Hadim, N. Mohamed, "Middleware: Middleware Challenges and Approaches for Wireless Sensor Networks", IEEE Distrib. Syst. Online, 2006.
[17] M. E. Hellman, "An Overview of Public Key Cryptography," IEEE Communications Society Magazine, vol. 16, no. 6, pp. 24-32, Nov. 1978.
[18] IEEE 802.15.4-2006 standard, http: // standards.ieee.org / getieee802 / download / 802.15.4-2006.pdf, 2015.
[19] S. Kaplantzis, "Classification techniques for network intrusion detection" , Technical Report, Monash University, 2004.
[20] C. Karlof, D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures", Ad Hoc Networks, vol. 1, pp.293 -315, 2003.
[21] I. Krontiris, T. Dimitriou, "Towards Intrusion Detection in Wireless Sensor Networks", Proceedings of 13th European Wireless Conference, Paris, 2007.
[22] B. Lai, S. Kim, I. Verbauwhede, "Scalable session key construction protocol for wireless sensor networks", IEEE Workshop on Large Scale RealTime and Embedded Systems (LARTES), 2002.
[23] C.E. Loo, M.Y. Ng, C. Leckie, M. Palaniswami, "Intrusion Detection for Routing Attacks in Sensor Networks", International Journal of Distributed Sensor Networks, 2006.
[24] S. Marchesani, L. Pomante, M. Pugliese, F. Santucci, "A Middleware Approach to Provide Security in IEEE 802.15.4 Wireless Sensor Networks," Proc. of the 6th International Conference on Mobile Wireless Middleware, Operating Systems and Applications, pp. 85-93, Nov. 2013.
[25] S. Marchesani, L. Pomante, M. Pugliese, F. Santucci, "Definition and Development of a Topology-based Cryptographic Scheme for Wireless Sensor Networks", 4th International Conference on Sensor Systems and Software (S-Cube 2013), Lucca, 2013.
[26] C. Mascolo, S. Hailes, "Survey of middleware for networked embedded systems", Technical Report for Project: Reconfigurable Ubiquitous Networked Embedded Systems,University College London, 2005.
[27] L. Pomante, M. Pugliese, S. Marchesani, F. Santucci, "WINSOME: A Middleware Platform for the Provision of Secure Monitoring Services over Wireless Sensor Networks", 9th International Wireless Communications & Mobile Computing Conference (IWCMC 2013), Cagliari, 2013.
[28] M. Pugliese, L. Pomante, F. Santucci, "Agent-based Scalable Design of a Cross-Layer Security Framework for Wireless Sensor Networks Monitoring Applications", Proceedings of the International Workshop on Scalable Ad Hoc and Sensor Networks (SASN2009), Saint Petersburg, 2009.

[29] M. Pugliese, L. Pomante, F. Santucci, “Secure Platform over Wireless Sensor Networks”, INTECH Publishers, ISBN 978-953-51-0218-2, 2012.
[30] M. Pugliese, F. Santucci, “Pair-wise Network Topology Authenticated Hybrid Cryptographic Keys for Wireless Sensor Networks using Vector Algebra”, Proceedings 4th IEEE Intern. Workshop on Wireless Sensor Networks Security (WSNS08), Atlanta, 2008.
[31] M. Pugliese, F. Santucci, “The Mean-Variance Estimator Technique in Monitoring Applications using Mobile Agents over Wireless Sensor Networks,” Proc. of the 6th International Conference on Mobile Wireless Middleware, Operating Systems and Applications, pp. 110-119, Nov. 2013.
[32] M. Pugliese, A. Giani, F. Santucci, “Weak Process Models for Attack Detection in a Clustered Sensor Network using Mobile Agents,” Proc. of the 1st International Conference on Sensor Systems and Software, pp. 33-50, Sep. 2009.
[33] R. Roman, J. Zhou, J., Lopez, “Applying Intrusion Detection Systems to WSNs”, IEEE Consumer Communications and Networking Conference, vol. 1, pp. 640–644, 2006.
[34] C.E. Shannon, “A Mathematical Theory of Cryptography,” Technical Report MM 45-110-02, Bell Labs., Sep. 1945.
[35] R. Sokullu, I. Korkmaz, O. Dagdeviren, A. Mitseva, and N. R. Prasad, “An Investigation on IEEE 802.15.4 MAC Layer Attacks”, Proc. International Symposium on Wireless Personal Media Communications, Dec. 2007.
[36] TinyOS homepage, http://www.tinyos.com, 2015.
[37] C. Vairo, M. Albano, S. Chessa, “A Secure Middleware for Wireless Sensor Networks”, Proceedings of the 5th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services, 2008.
[38] M. Valero, S. S. Jungy, A. S. Uluagacy, Y. Li, R. Beyahy, “Di-Sec: A Distributed Security Framework for Heterogeneous Wireless Sensor Networks”, Proceedings of the IEEE INFOCOM Conference, 2012.
[39] M. Wang, J. Cao, J. Li, S. K. Das, “Middleware for Wireless Sensor Networks: A Survey”, J. Comput. Sci. Technol., 2008.
[40] Y. Xiao, S. Sethi, H.-H. Chen, B. Sun, ”Security services and enhancements in the IEEE 802.15.4 wireless sensor networks”, in Proceedings of IEEE GLOBECOM’05, vol.3, 2005.
[41] W. Xu, K. Ma, W. Trappe, Y. Zhang, ”Jamming sensor networks: attack and defense strategies”, IEEE Network, vol.20, no.3, pp. 41-47, Jun. 2006.
[42] Q. Zhang, T. Yu, P. Ning, “A Framework for Identifying Compromised Nodes in WSNs”, ACM Transaction Information System Security 11(12), 2008
[43] ZigBee Alliance: ZigBee Document 053474r17, ZigBee Specification. ZigBee Alliance, Jan. 2008.