Computer Networks 199 (2021) 108470

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Lightweight method of shuffling overlapped data-blocks for data integrity and security in WSNs

Francisco Alcaraz Velasco, Jose Manuel Palomares *, Joaquin Olivares

Department of Electronic and Computer Engineering, Universidad de Córdoba, Córdoba 14071, Spain

ARTICLE INFO

Keywords:
Wireless sensor networks
Security
Shuffling
Integrity
Overlapping blocks

ABSTRACT

Wireless Sensor Networks *(WSN)* consist of devices with limited resources to explore and sense the environment in a cooperative way. Security, mainly in terms of guaranteeing the data integrity, is a primary issue for many applications, but with an extra energy cost. Thus, trade-off is required between security level and energy consumption in real applications. First of all, a brief survey about security methods, focusing in data integrity, in *WSN* is implemented. The objective of this paper is to propose a new data integrity method with medium security levels and low energy cost. Therefore, we propose a new and lightweight mechanism for data integrity with overlapping blocks in *WSNs*. Hence, an attacker will spend much time and effort to interpret and alter the packets. The experiments were performed using *TinyOS 2.1* operating system and *TelosB* nodes for measuring the overhead in terms of energy consumption, memory, and packet size. Moreover, the receiver is able to detect tampering packets and request those retransmission data. An attacker would require huge amounts of memory and processing time to extract the original information, even for small-sized data blocks. Thus, this fact makes this approach a simple, yet effective, mechanism to protect data whilst enhancing the data integrity.

1. Introduction

Wireless Sensor Networks (WSN) consist of small and autonomous devices wirelessly connected cooperating to monitor or control an area. These devices are based on small microcontrollers, low power radio transceivers, limited power supply, sensors, and actuators. Thereby, they present some limitations, as energy availability, computing power, bandwidth and coverage to keep power consumption low [1,2].

Nevertheless, these networks create intelligent services to make living environments more comfortable and safer with the networked interconnection of everyday objects, in what is known as Internet of Things [3]. In this way, wireless sensor networks are increasingly becoming more widespread.

However, WSNs collect, transmit, store and potentially share vast amounts of consumer data, and some of them contain sensitive and private personal information. Therefore, mechanisms need to be used to send the information securely [4,5], as security is an important issue in WSNs [6,7]. However, the sensor nodes have limited hardware resources and power energy units, and therefore there is a trade-off between security and computing power [8].

Current trends in WSN and IoT usually require large amounts of redundant data obtained from many sources, mainly sensors. Although sensors send the data individually, new works [9,10] are providing frameworks to send efficiently only the relevant information, structured in streams of blocks of data. In these cases, the security of each single data is not so important, as the final systems are able to handle properly some missing or altered data. The final systems must receive huge amounts of data to perform the decision making. Therefore, those nodes hosting the final systems are designed with buffers big enough to store all the received data required for the computation. Thus, the security for these systems can be considered within the integrity of each individual data inside a larger block of information.

In this sense, this work aims at proposing a simple yet secure mechanism for multiple data flows. Therefore, our scope is focused in nodes sending multi-packet data messages composed by a mixture of several signals. So, it is possible to apply data combination to build a secure data block. However, current state-of-the-art security mechanisms do not seem to be suitable for the low-power, high-constraint wireless motes involved in most WSNs. For instance, security solutions based on encrypt algorithms would introduce large computational overhead. Techniques based on watermarking are not able to make data flows unintelligible. Besides, these solutions do not protect against side channel attacks [11]. Therefore, we propose a distributed and light security scheme which prevents data blocks against tamperings. Moreover, the proposed method transforms the interchanged data flows to make them almost unintelligible to illegitimate users.

* Corresponding author.
E-mail address: jmpalomares@uco.es (J.M. Palomares).

https://doi.org/10.1016/j.comnet.2021.108470
Received 20 January 2021; Received in revised form 24 April 2021; Accepted 6 September 2021
Available online 13 September 2021

The rest of this article is organized as follows. Section 2 briefly introduces techniques related with the data integrity and confidentiality in WSNs. Section 3 describes our overlapping block mechanisms proposed to protect WSNs from attacks and misbehavior. Section 4 presents the hardware and software platform used to develop the proposals. Section 5 describes the experiments and measures to evaluate our methods. Section 6 evaluates our proposals from the point of view security services. Section 7 shows a comparison of the resource consumption of the proposed scheme and other techniques presented in Section 2. Section 8 concludes the technical report.

2. Foundations

Integrity and confidentiality are the leading security requirements in WSNs to protect the information and resources from attacks and misbehavior [12]. In this section, some works are described. The advantages and disadvantages of each technique are also highlighted.

2.1. Integrity schemes

Wireless communications are highly affected by external interactions which can modify the messages. These modifications may be unintentional (electromagnetic interference) or intentional (as injection attacks or alteration of messages) [6]. Consequently, some mechanisms are needed to detect errors or changes in messages. These mechanisms add redundant or additional information, called *Frame Check Sequence (FCS)*, to the message to detect errors in transmission. In the following sections, we present several techniques based on error-detecting codes, hash functions, cryptography algorithms or watermarking schemes to get *FCS* values.

2.1.1. Error-detecting codes

Error-Detecting Codes are widely used either in wired or wireless networks. Only frames marked as correct are delivered to upper layers of communication protocols, whilst frames with errors are dropped. *Cyclic Redundancy Check (CRC)*, one's and two's complement addition checksums are some examples of error-detecting codes.

Cyclic Redundancy Check (CRC): it is a powerful technique to detect transmission errors. Unlike the previous schemes, which are based on addition, CRC is based on binary division. *CRC-12*, *CRC-16* and *CRC-CCITT* are some examples of standardized polynomials.

Koopman at el. [13] evaluate the comparative error detection effectiveness, as well as cost performance trade-off points, of the most commonly used checksum approaches in embedded networks: XOR, two's and one's complement addition, Fletcher and Adler checksum, and CRC. They conclude that for all networks, a good CRC polynomial, whenever possible, should be used for error detection purposes.

2.1.2. Hash functions

A hash function H maps an input message M to a fixed-length output $H(M)$. Any modification to the original message M will result in a different hash value. This property enables the verification of integrity messages. The followings hash functions have been analyzed:

- *MD5*: *Rivest* [14] obtains a hash length of 128 bits. *Kyoungsoo et al.* [15] propose a secure scheme named *TinyMD5* which uses a modification of *MD5* to generate a 32-bit hash. Due to a one-way hash function, it has no decryption algorithm, the original data are recovered by data matching on the base station. The recovering process uses a matching and finding process in the hash values stored in a database.
- *SHA1*: *Secure Hash Algorithm (SHA)* was one of the most widely used hash functions.

2.1.3. Message authentication code approaches

Message Authentication Codes (MACs) are a block of a few bytes that is used to authenticate and check the integrity a message. Sender and receiver share a secret key. Different functions or algorithms can be used to calculate *MACs*, some of them are:

- *HMAC*: *Keyed-Hashing for Message Authentication*, defined in *RFC 2104*, it can be used with any iterative cryptographic hash function like MD5 or SHA-1, along with a secret shared key.
- *Cipher-Block Chaining (CBC-MAC) Mode*: to get the MAC code, the message is encrypted using a cipher block operating with *CBC* mode. In *TinySec* [16], link layer security architecture uses CBC as its default mode of operation and *Skipjack* [17] as the encryption algorithm. According to the experiments carried out by *Jongdeog et al.* [18] on *TelosB* and *AES* with a 128-bit key as encryption algorithm for 16-byte messages, CBC-MAC, XMAC and CMAC take, *15.02* ms, *23.61* ms and *33.51* ms, respectively. *Pereira* [19] analyzes the *Marvin* message authentication code with *Curupira-2* as encrypt with 96-bit key size and 12-byte *MAC* size. On *TelosB*, it requires *27.16* ms computation time for 108-byte messages.
- *Offset Codebook Mode (OCB) approaches*: *TeenySec* [20], a data link layer security protocol is based on OCB and a private encryption algorithm, *Corrected Block Tea Algorithm (XXTEA)*. This protocol outperforms *Tinysec* [16] because it consumes less energy and produces less overhead in the packets.

2.1.4. Watermarking approaches

Techniques based on watermarking offer lightweight data integrity and authentication schemes for WSN. The main objective of these techniques is to introduce a piece of secret information, namely a watermark, to detect any change in the original data streams. *Kamel and Hussam* [21] propose the MD5 hash function to generate the watermark. Then, at the sender side, the watermark is embedded in the least significant bits of the data.

In [22], a *Position Random Watermark (PRW)* method is designed. The digital watermark is generated by a SHA-1 hash function and only some from the most significant bits of the watermark are embedded into dynamic positions of each packet.

However, watermarking methods are not comparable with our proposed method because an attacker could interpret the transmitted data. Also, the computational cost to calculate only the hash value is high. Besides, most watermarking methods require redundant bits in the data reading to embed the watermark, so it could be a weakness, if the underlying WSN does not support those distortions in the data reading.

2.2. Confidentiality approaches

In this section, some techniques to prevent the disclosure of secret and private information are analyzed. These mechanisms are classified in two categories: symmetric and asymmetric key cryptography algorithms [23]. Energy consumption, time execution, and memory consumption are parameters evaluated to obtain the algorithm performance.

2.2.1. Symmetric key encryption

These algorithms need a single key to encrypt and decrypt. Due to the limited resources of WSNs, choosing any algorithm is a great challenge. *Cazorla et al.* [24] present a study of 21 encryption blocks used in the *Texas Instruments* 16-bit *MSP430* microcontroller. Concerning performances, the best encryption/decryption processes in cycles/bytes are *AES, Noekeon, SPECK64, SPECK128 and SEA* algorithms, which require less than 1000 cycles/byte.

Jongdeog et al. [18] studied *AES, RC5, Skipjack*, and *XXTEA* encryption algorithms on *TelosB* and *MicaZ* motes. Using *Cipher block chaining (CBC)* as mode of operation: *Skipjack* is the most energy-efficient algorithm of the fours.

Table 1
Original non-securized method.

Message-1	FCS-1-Hw
Message-2	FCS-2-Hw
...	..
Message-n	FCS-n-Hw

Table 2
Overlapping Blocks, (OB).

Msg 1-a	Msg 2-a	..	Msg n-a	FCS-a-Sw	FCS-Hw
Msg 1-b	Msg 2-b	..	Msg n-b	FCS-b-Sw	FCS-Hw
..	..	..	..	..	FCS-Hw
Msg 1-m	Msg 2-m	..	Msg n-m	FCS-m-Sw	FCS-Hw
FCS-1-Sw	FCS-2-Sw	..	FCS-n-Sw	FCS-m-n-Sw	FCS-Hw

Table 3
Overlapping Blocks with Horizontal Shuffle, (OB_Ho).

Msg n-a	FCS-a-Sw	..	Msg 1-a	Msg 2-a	FCS-Hw
Msg 2-b	Msg 1-b	..	FCS-b-Sw	Msg n-b	FCS-Hw
..	..	..	..	..	FCS-Hw
Msg n-m	FCS-m-Sw	..	Msg 1-m	Msg 2-m	FCS-Hw
FCS-2-Sw	FCS-1-Sw	..	FCS-m-n-Sw	FCS-n-Sw	FCS-Hw

2.2.2. Asymmetric key encryption

These algorithms use a pair of different keys. Although private key algorithms are much faster than public key algorithms, they present some weaknesses such as: key management, key life-time, key interchange and digital signature [23].

Utku at al. [25] present an efficient software implementation of the *Rivest–Shamir–Adleman (RSA)* cypher algorithm on *TelosB* devices, utilizing several acceleration techniques. It achieves 1024-bit RSA encryption and decryption operations on MSP430 in only 0.047 s and 1.14 s, respectively.

Elliptic curve cryptography (ECC) is another widely regarded public key cryptographic primitive. Successive improvements in the math implemented in *TinyECC* [26] on *TinyOS* have allowed us to get a 2*s* running for signing and verifying.

3. Methodology

In this section, we describe three overlapping block mechanisms which provide three security levels. Furthermore, it is possible to choose one mechanism according to the security level required by the application. Table 1 shows the scheme to send a data block and the *Frame Check Sequence (FCS)* which is calculated by the hardware radio *(FCS-Hw)*. In Section 3.2 the structure of the messages is described with more detail.

Table 2 shows the first proposal, which we have named *Overlapping Blocks (OB)*. In the first step, the transposed matrix is generated with data block, and, in the second step, the *FCS* is calculated via software for each column and row of the data blocks. The integrity software method to calculate the *FCS* will be discussed in Section 4. Therefore, the n messages sent in the original non-securized method are sent in the OB scheme by dividing each original message into m different pieces and creating a new message which is a composition of each different part of the original messages. Thus, the original messages can be obtained only if all the messages are captured.

The second method is named *Overlapping Blocks with Horizontal Shuffle, (OB_Ho)*. It is shown in Table 3. In this case, the chunks are shuffled among columns but maintaining their position in rows. Therefore, it is harder to interpret the communication for an intruder. Graphically, the columns have different colors with respect Table 2.

Table 4 shows the last method named *Overlapping Blocks with Complete Shuffle, (OB_HoVe)*. The chunks can be shuffled among columns and rows. Thus, without the prior knowledge of the ordering, it is much more complex to interpret the messages by an attacker. Table 4 shows an example after shuffling a data block.

Table 4
Overlapping Blocks with Complete Shuffle, (OB_HoVe).

FCS-1-Sw	Msg 2-m	..	FCS-b-Sw	Msg n-a	FCS-Hw
Msg n-m	FCS-a-Sw	..	Msg 1-b	Msg n-b	FCS-Hw
..	..	..	..	..	FCS-Hw
Msg 2-b	Msg 2-a	..	Msg 1-m	FCS-m-n-Sw	FCS-Hw
Msg 1-a	FCS-n-Sw	..	FCS-2-Sw	FCS-n-Sw	FCS-Hw

The receiver, using the software-computed *FCS-Sw* values, is able to detect transmission errors and may request data retransmissions. The receiver signals only the rows and columns where the *FCS-Sw* values present errors. This procedure reduces energy consumption because a complete resending of the complete data block is unnecessary.

3.1. Network architecture

In this section, the network architecture assumed in our protocol is described. This architecture is based on clustering [27]. The cluster head must first check data security. After that, it shuffles the received data, and finally, sends the processed data to base station.

In Fig. 1, this architecture is presented. Nodes send *Data Flow* to the head cluster, named *D1, D2, D3*. The head cluster builds a *Data Message* after applying a data combination method. Then, *Data Message* is shuffled and *FCS* integrity control is calculated. After that, a *Data Block* is built with these data messages, which are sent using as many packets as necessary to send all the information to the Base Station.

3.2. Packet and message structure

Fig. 2 shows the structure of the *Data Messages* of *OB* method, which are encapsulated inside of the *Data Packets*.

The fields of the *Data Packets* structure depends on the communication standard. For instance, if the communication standard is *IEEE 802.15.4* [28]:

- *Metadata*: example of these fields are frame control and addressing fields.
- *Data Payload*: this field encapsules the application data.
- *CRC-Hw*: this field is used for integrity control, which is calculated by the modem.

The *Data Message* fields are the following:

- SN-Data: *Sequence Number-Data*, this field has two objectives: The first to provide freshness data control. This prevents an intruder from requesting copies of old data values. The second one is to provide authentication. The Sender and head-cluster must be synchronized with the number of blocks that have been received and sent. Using this value, the first byte value of a data packet is decreased by the sender, and then it is light-signed using the scheme proposed in [27]. The head cluster performs the inverse process. This scheme has as an advantage that it does not generate any additional overhead.
- Data: In this field, data readings of the OB mechanism are sent.
- FCS: *Frame Check Sequence* is computed via software. In Section 4, the algorithm will be selected, based on our tests. Its objective is to detect security faults and errors.

3.3. Authentication and seed generation

The working architecture of the proposed method is the clustering depicted in Fig. 1. Thus, the Data Block will be authenticated using the scheme proposed by Bouakkaz et al. [27].

The head cluster and nodes must carry out a handshake process to interchange data of our overlapping mechanism. These sensitive

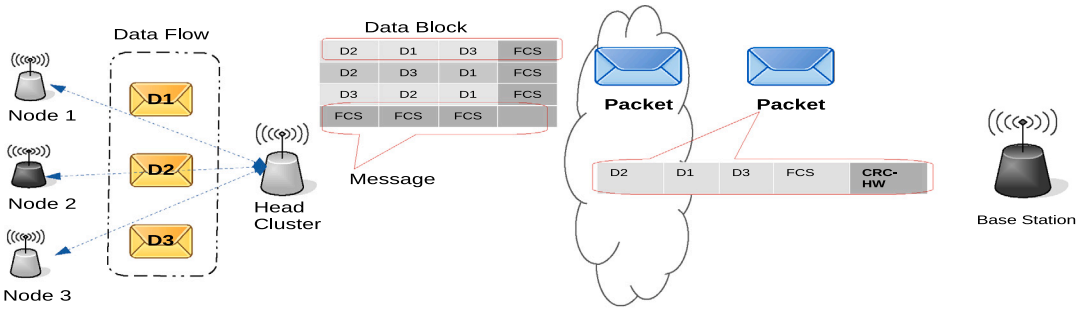

Fig. 1. Network Architecture.

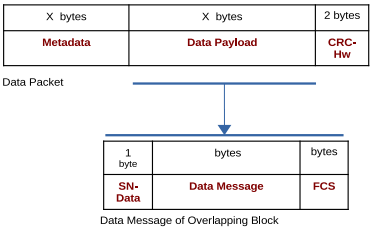

Fig. 2. Packet and Message Structure.

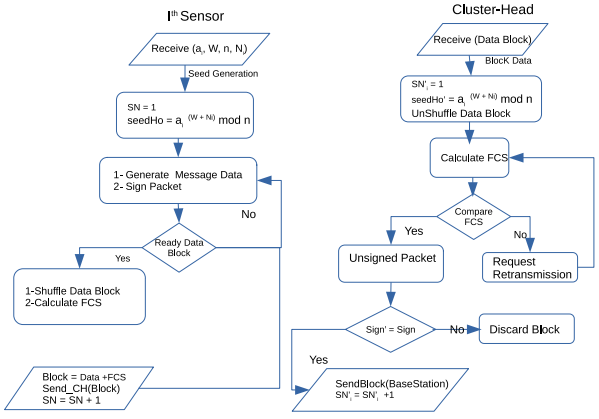

Fig. 3. Procedure for sending information from a node to the Head cluster using Overlapping Block method.

data should be securely sent. We use a private cypher mechanism and, we take advantage of the radio [29]. Therefore, the head cluster generates an N_{hc} nonce value as random value, an increasing sequence $A = (a_1, a_2, \ldots, a_t)$, with t value as the cluster size. After that, the head cluster calculates two integer values *(W, n)*, where the n value is greater than the sum of the a_i values and the W value must satisfy $gcd(W, a_i) = 1$. Then, the head cluster sends each different private-share key (a_i, W, n, N_{hc}) to each node. Besides, during the shuffle process the seed value is also calculated using this scheme.

The head cluster must be synchronized with each $node_i$ to know how many blocks have been sent by each node. The head cluster stores that value in the sequence $SN = (sn_1, sn_2, \ldots, sn_t)$. Fig. 3 shows the execution flow diagram with the main steps for signing and seed generation process from the head cluster to a $node_i$. The sender node signs the packet according to the following equations:

$$Sign = a_i^{(W+SN+N_{hc})} \mod n \tag{1}$$

$$SN\_Data = (SN\_Data - Sign) \mod 255 \tag{2}$$

The head cluster checks the authenticity of a data message using Eq. (3). The integrity control will fail if the first byte of data message has been altered or a replay attack has been generated.

$$SN\_Data = (SN\_Data + Sign) \mod 255 \tag{3}$$

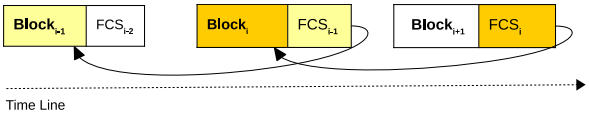

Fig. 4. FCS Backwarding.

3.4. Shuffle algorithms

Algorithm 1 shows the method to shuffle the chunks using the *Overlapping Blocks with horizontal shuffle* scheme *OB_Ho*. The algorithm used to apply *Overlapping Blocks with complete shuffle* scheme, *OB_HoVe* is similar to *OB_Ho*, but it adds shuffle between rows.

Algorithm 1 Overlapping Blocks with horizontal shuffle

```
1:  procedure EXCHANGE BY COLUMNS(BlockData, seedHo)
2:      rows ← lengthPacket
3:      columns ← numberPacket
4:      seed ← (seedHo + numBlockSent)
5:      for i ← 0, rows do
6:          for j ← 0, columns do
7:              jump ← (j + i + seed) mod columns
8:              if block(j) and block(jump) not shuffled then
9:                  aux ← block(j)
10:                 block(j) ← block(jump)
11:                 block(jump) ← (aux)
12:             end if
13:         end for
14:     end for
15: end procedure
```

To increase the security level, the seed value is increased after sending each block. After that, the seed value is added to each block position of the data block chunk to obtain different shuffling values.

3.5. FCS backwarding

This section explains how *FCS* values are computed for the columns of block B_i. These values are sent along with block B_{i+1}. With this method, the security level is increased because the correlation between *FCS* and the data blocks is reduced. However, this method has a small drawback, because both sender and receiver need larger buffers. This functionality is depicted in Fig. 4.

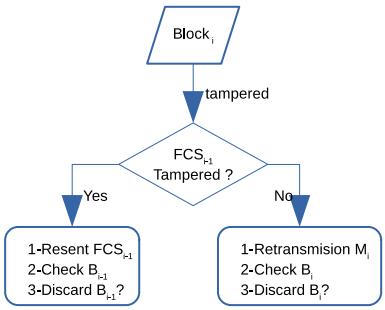

Fig. 5. Countermeasures when B_i block is tampered.

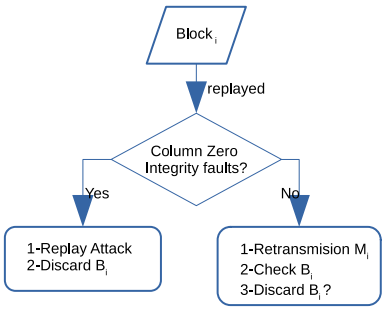

Fig. 6. B_i block is replayed.

3.6. Attack model and security analysis

WSN are exposed to more threats than wired networks, because they have limited resources such as low available computational resources and energy. Besides, the security protocol is also more complex for those reasons. In this section, four usual types of attacks are analyzed, and how our proposed method can prevent and mitigate their effects.

3.6.1. Attack model

Diaz et al. [30] make a classification of the type of attacks in *WSNs*. We focus on four important and common attacks, which are the following:

1. Tampering packet: a malicious node alters some data packets and then, it forwards them.
2. Replay attack: an intruder resends old data packets. Therefore, the freshness of the data is affected.
3. Packet forgery: a malicious node sends fake data packets, so the traffic network is increased and the energy consumption rises, consequently.
4. Selective forwarding: a malicious node sends data packets selectively and other can be partially deleted, therefore this may cause a loss of data in the head cluster.

3.6.2. Security analysis

This section analyzes how our protocol should behave when an attack is presented. Fig. 4 depicts a data block sequence together with the *FCS Backwarding* scheme. The following notation is used: B_i is the current data block, FCS_{i-1} is the integrity control by columns for the B_{i-1} block and B_{i+1} is the following data block. M_i is a data message, which belongs to the B_i block and its structure is depicted in Fig. 2.

1. Tampering packet: if the B_i block has been tampered, the following may happen:
   - The modifications affect some bytes in the FCS_{i-1}. An integrity error of B_{i-1} block is thrown. It is detected because the horizontal integrity will be corrected despite the fact that the vertical integrity check will fail; therefore, retransmissions are requested for resending the FCS_{i-1} values where there are column errors. If the maximum amount of attempts is reached, the B_{i-1} block is discarded.
   - The modifications affect bytes in the M_i data message of the B_i block. When the head cluster receives the B_{i+1} block, an integrity error will occur. Therefore, the head cluster will proceed to request retransmissions or to discard B_i block if the maximum amount of attempts is reached.

   Fig. 5 depicts in a diagram how the B_i block is tampered.
2. Replay Attack: if a replay attack is launched in the B_i block, some M_i data messages become old. The head cluster will detect it when it receives the B_{i+1} block, because the integrity control of zero column will fail. However, the integrity control by rows may be right, it would provoke the complete resubmission of the B_i block. Thus, the B_i block is discarded. This behavior is represented in Fig. 6.

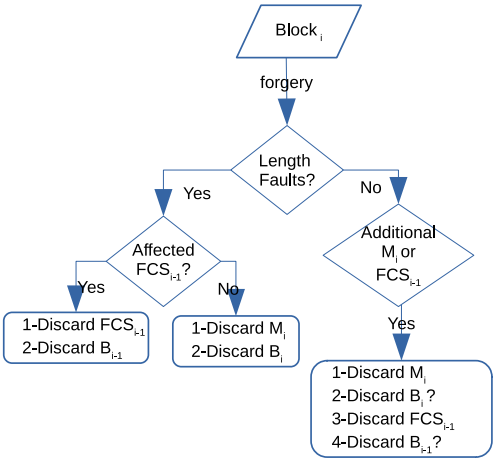

Fig. 7. B_i block is forged.

3. Packet forgery: the B_i block is altered by the inclusion of additional data. Fig. 7 depicts this procedure. The following options are considered:
   - The additional bytes change the length of the M_i or FCS_{i-1} making the message to be incorrect. So, the head cluster discards the B_i block in the first case (incorrect Mi length) and the B_{i-1} block, in the other case.
   - Another case comes out when those additional messages are inserted into the B_i block. So, when the number of messages received achieves the size of block, all additional messages are discarded. This, may provoke a fault in an integrity control of the B_i block because values of FCS_i may not match. So, the B_i block should be discarded.
   - Additional messages which contain the values of the FCS_{i-1} of the columns are discarded. However, in this case, the integrity control of the B_{i-1} block may be affected. So, B_{i-1} should be discarded.
4. Selective forwarding: the B_i block is forwarded by deleting of some data values. Fig. 8 shows the flow diagram of this behavior. The following options are considered:
   - Some readings of message M_i are deleted, therefore the length of the message is incorrect. So, this M_i message is discarded by the head cluster and retransmissions are requested. If the maximum amount of attempts is reached, the B_i block is discarded.

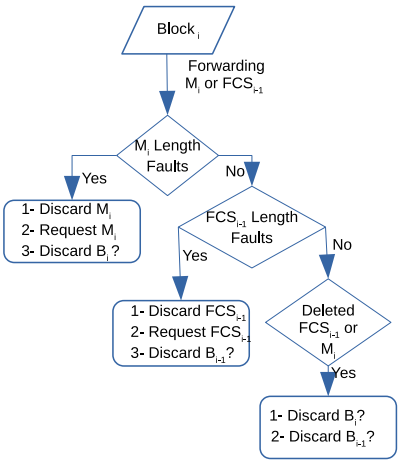

Fig. 8. B_i block is forwarded.

- Second case, some readings of messages which contain the values of the FCS_{i-1} are deleted. Therefore, their lengths are incorrect and, these messages are discarded. Once again, retransmissions are requested. If the maximum amount of attempts is reached, the B_{i-1} block is discarded.
- Another option is that M_i or FCS_{i-1} messages are fully deleted. The B_i block is discarded in the first case, and the B_{i-1} block is rejected, in the second case.

4. Implementation

In this section, we present the hardware and software platform used in this project.

4.1. Execution environment

The following execution environment is used to implement our mechanisms and to obtain experimental results and measures:

- The *CM5000* mote is an IEEE 802.15.4 [28] compliant wireless sensor node based on the original open-source *TelosB* platform [31]. We have selected *TinyOS* as open source Operating System, which is commonly used by the scientific community. The implementation of the proposals have been written using *nesC* programming language [32], as a set of cooperating tasks and processes.
- Moreover, we have used a second environment to study the *Hamming Distance* and the *Recall* metric. This environment is completely simulated and has been programmed in *Java*. It uses the Eclipse IDE to provide the interface to obtain the results.

4.2. CBC mode CC2420

The IEEE 802.15.4 standard provides security at the link layer, supported by hardware in most standard radio transceivers, such as the *CC2420* used on *TelosB* motes. This radio is able to perform encryption and decryption using the 128-bit *Advanced Encryption Standard (AES)* algorithm supported by IEEE 802.15.4. According to the IEEE 802.15.4 standard, there are three modes of operation for encryption: *Cipher block chaining-message authentication code (CBC-MAC)*, *Counter* (CTR) and *Counter with CBC-MAC* (CCM-MAC).

At first, we focused our attention in the *CBC-MAC* mode. When a message is received, a *Message Integrity Code (MIC)* is computed in the first step. After that, in a second step, the computed MIC value obtained from the message is compared with the *MIC* value stored in the *RXFIFO* register. The last byte of the *MIC* is replaced by hardware. Therefore, the original *MIC* value is not accessible to the OSI layers of the network stack model. Then, the *Counter with CBC-MAC* (CCM-MAC) works with the *MIC* value as with the *CBC-MAC* mode. Thirdly, the *Counter* (CTR) does not provide integrity control. Due to this, these modes of operation cannot be used to control the integrity of data messages in our proposed method.

Table 5
Execution time and overhead length.

Checksums	Time	Overhead
XOR	4.43 µs	16 bits
C1	4.53 µs	16 bits
C2	4.37 µs	16 bits
ITU-T CRC-16	4.98 µs	16 bits
Hashs	**Time**	**Overhead**
MD5	1.73 ms	128 bits
SHA1	6.2 ms	160 bits
MAC-algorithm	**Time**	**Overhead**
HMAC	92.75 ms [19]	128 bits
Marvin	17.54 ms [19]	96 bits
CBC-MAC	15.7 ms [18]	32 bits

4.3. CRC-16 computation by CC2420

The *CC2420* radio computes a 2-byte frame check sequence (FCS) following the last *MAC* payload byte, using the *ITU-T CRC-16*. When a packet is received, the most significant bit in the last byte of each frame is set to high, if the *CRC* of the received frame is correct, and to low, otherwise. The remaining 15 bits of *FCS* are replaced by the *RSSI* value, which is the average correlation used for *LQI*. Therefore, the original value of *CRC16* cannot be queried by the upper layers of the network stack. Thus, the *CRC-16* computation by *CC2420* cannot be used in our proposal.

4.4. Selected frame check sequence

In this section, the *Frame Check Sequence* of our mechanism is selected among the alternatives presented in Section 2. The studied parameters are: execution time, overhead length, and Hamming weight. Table 5 shows the execution time and the overhead length. *Checksums* and *Hash* algorithms have been programmed in *nesC* on *TinyOS* Operating System. The processing times have been measured through *LocalTimeMicroC* components of *TinyOS*. Execution times of *MAC* algorithms have been taken from [19] and [18].

In Maxino et al. [13] the checksums using the *Hamming weights* of *XOR*, *C1 and C2* complements checksums are studied.

We decided analyze the different checksum algorithms according to the *Hamming Weights* in order to select the most suitable one for our proposal. Some of the values selected for the analysis are the following: 16 bits are used for the integrity control and 32 bytes for length messages. Also, 2, 4, and 6 errors in different random positions have been injected. According to Fig. 9, the *CRC-16* of *CCITT* provides the best results compared to the *XOR, C1* and *C2* complements, because it has the lowest *Hamming weights* of all of them.

Therefore, we have decided to use the *CRC-16* of *CCITT* for the computation of the *Frame Control Sequence* to implement our security mechanisms, as proposed in Section 3.

5. Results

In this section, we described the experiments (or benchmark tests) about our *OB* mechanisms. We included a simulated man-in-the-middle module, which randomly changes some bytes of the data block to verify that the head cluster is able to detect these modifications and requests the sender to resubmit all the altered data.

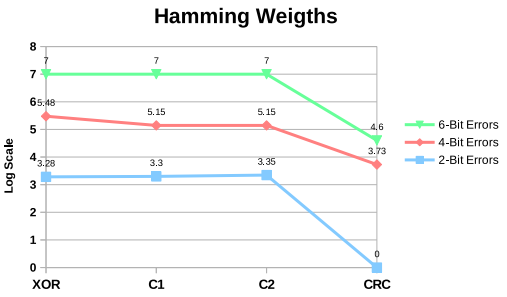

Fig. 9. Hamming weights.

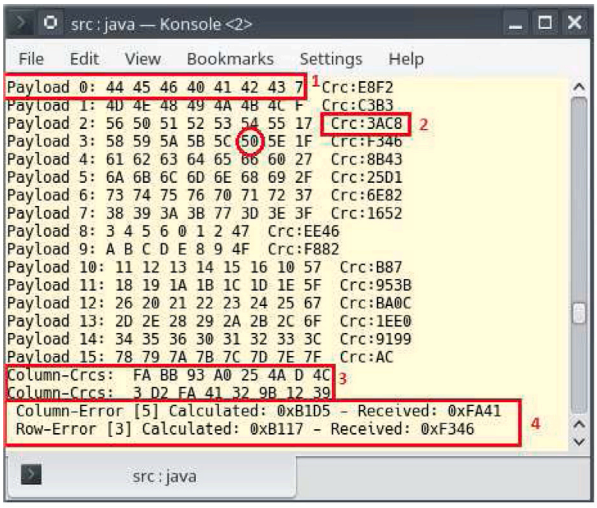

Fig. 10. Receiving Data with error.

5.1. Simulated man-in-the-middle

This section shows how two motes send a 128-byte data block, with data values going sequentially from 0 to 127, in order to help in the evaluation of the results of the experiments. The motes use the *OB* mechanism. Besides, a simulated man-in-the-middle module is included. This module is programmed to alter 1 byte with a 50 hexadecimal value.

In Fig. 10, a data block is sent from Node 2 to the Head Cluster (Node 3). Some labels have been included in Fig. 10. In the following list, every labels is described:

1. Payload in hexadecimal notation.
2. *CRC-16* computed for a row.
3. *CRC-16* computed for columns of data block.
4. The Head Cluster (Node 3) detects one error in the fifth column and third row. Therefore, it requires Node 2 to resend those data.

Fig. 11 shows the result of a test where:

1. Node 3 sends to Node 2 a *NO_ACK_BLOCK* message because an error is detected when comparing the *CRC* values. Besides, Node 3 sends the column and row numbers where the error has been detected. These errors are represented in Fig. 10 with a highlighted 50 hexadecimal value.
2. Node 2 resends only the data corresponding to the intersection of the columns with the rows where there are errors.
3. If no error is detected, Node 3 informs Node 2 with an *ACK_BLOCK* message. After that, Node 3 shuffles back the data block using Algorithm 1 to obtain the original position of data block.

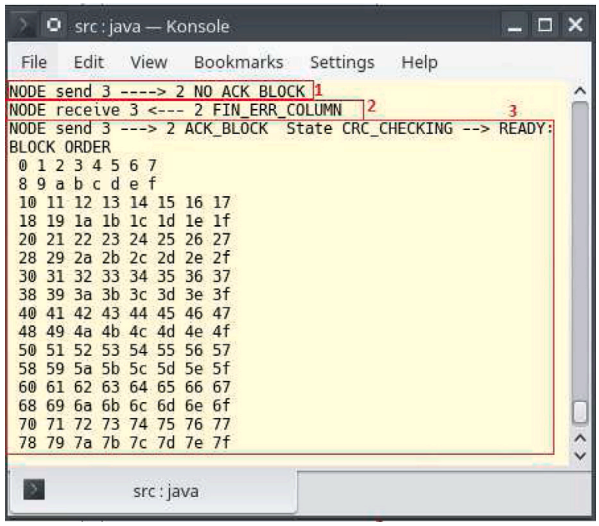

Fig. 11. Detecting Errors.

Table 6
Overhead data messages (bytes).

	Data	Overhead	% Overhead
base	128	0	0%
OB	128	48	37.5%
OB_Ho	128	48	37.5%
OB_HoVe	128	48	37.5%

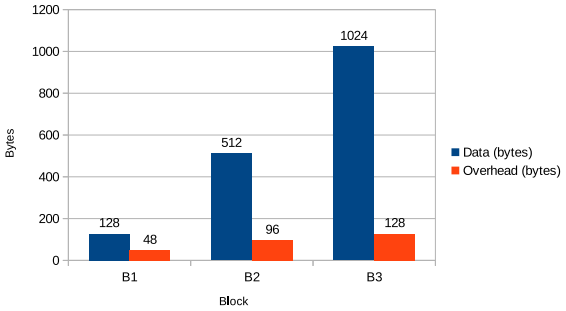

Fig. 12. Example Overhead Packets.

5.2. Overhead

In this subsection, the overhead from the four alternatives is shown: first, the data block is sent without any security solution, this version is called *base*. In the second option, integrity control is added with our (OB) method which adds the integrity control for each row and column of data block. Third alternative is added with the horizontal shuffle mechanism *(OB_Ho)*, and the last one is with all the previous options and complete shuffle *(OB_HoVe)*. The data block has a size of 8 bytes of payload and 16 messages, so the block has a total length of 128 bytes. Table 6 shows the results. There is no overhead increase among *(OB)* mechanisms. The main overhead element is due to the integrity control. This value decreases if the block size is greater. In Fig. 12 three different size blocks are shown. The third block, *B3*, has a size of 1024 bytes and requires 128 bytes to send all *CRCs*, so the overhead is 12.5%, which is lower than the one in block *B1*, which is 37.5%. because all the computation to shuffle carried out by each node and the head cluster is the same.

Table 7
Transmission Time (μs), Time increase(%), Energy Consumption (mJ), and Energy increase(%).

	Time	% Time	Energy	%Energy
Tiny	54.78	0	31.46	0%
OB	68.93	20.5%	39.54	20.43%
OB_Ho	88.24	37.9%	50.67	37.91%
OB_HoVe	90.52	39.5%	51.98	39.47%

Table 8
ROM/RAM usage memory (bytes).

	ROM	RAM	%ROM	%RAM
Tiny	18,336	2,752	36.5%	22.86%
OB	18,780	2,896	37.4%	24.05%
OB_Ho	19,022	2,896	37.9%	24.05%
OB_HoVe	19,330	2,912	38.5%	24.19%

5.3. Energy consumption

This section shows the transmission time and energy consumption based on *TelosB* nodes. The results are shown in Table 7. The following assumptions are considered:

- The size of data block is 128 bytes.
- The current consumption of the radio modem to transmit is 17.4 mA with a supply voltage of 3.5 V.
- The average time spent (μs) to send five blocks.

5.4. Usage memory

In this section, memory usage is analyzed. Table 8 shows this usage with other security solutions. The following considerations have been taken into account:

- The size of the block is 16 columns and 8 rows, therefore 128 bytes are sent.
- The first version, called *Tiny*, sends the data blocks without any security.
- Security services have been included incrementally: authentication service and integrity control in the *OB* method. The horizontal shuffle is included in the *OB_Ho* method. Finally, *OB_HoVe* is included with full horizontal and vertical shuffle.
- The *TelosB* nodes have 49 KB ROM memory and 10 KB RAM memory.

The percentage of usage introduced by the *OB_HoVe*, which has all security services, produces an overhead of 1.23% RAM as compared to the *Tiny* version and 2.0% in ROM. Therefore, this overhead has a very low impact in comparison with unsecured version.

5.5. Recall

In this subsection, the recall values are presented. 200 data blocks have been used, with a size of 8 messages and a payload of 16 bytes. Besides, all blocks have some collision in the checksums. Therefore, these errors cannot be detected by the *XOR, C1, C2* and *CRC16* methods. However, our *OB* methods include a double integrity control, which provides a 100% recall, as it is shown in Table 9.

5.6. Retransmissions

In this section, the percentage of retransmitted bytes and the savings when an error is detected are presented. The following issues must be taken into consideration:

1. Due to the mathematical properties of *CRC*, when an error is detected, the rows and columns which were modified are known.

Table 9
Recall values.

	False negative	True positive	Recall
XOR	200	0	0%
C1	200	0	0%
C2	200	0	0%
CRC16	200	0	0%
OB	0	200	100%
OB_Ve	0	200	100%
OB_HoVe	0	200	100%

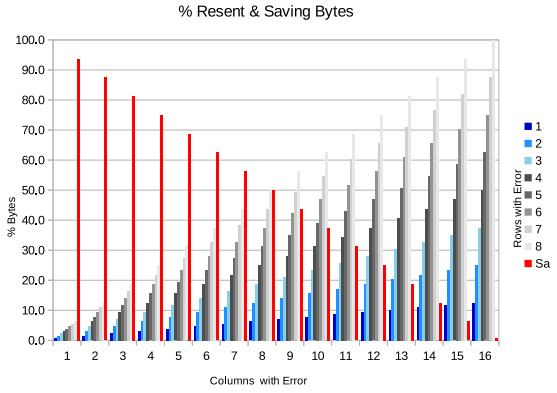

Fig. 13. Percentage of Resent and Saved Bytes.

2. Considering R_c, R_r the two different identifiers of columns and rows with errors, the amount of retransmitted bytes are $R = R_c \times R_e$. Therefore, this number is a suboptimal value due to issue number 1.
3. The worst case occurs when the modified values are the only ones in the diagonal block. This triggers the resubmission of a quantity between 50% and 100% of the blocks, depending on the size block.

In Fig. 13, a block with 16 bytes of message length and 8 messages by block is used for the experimental setup. Therefore, the size block is 128 bytes. This figure represents the percentage of retransmitted bytes versus the percentage of saved bytes because the block has not been completely resent. These savings are represented in red under the series named *Sa (Savings)*. Savings ranging from 80% to 90% when R_c is low and R_r is high, or viceversa have been obtained. It is worth mentioning that when the savings are below 50%, the best option is to discard the block completely and request a complete retransmission.

5.7. Throughput

This section shows the throughput using three block sizes: 64, 128, and 256 bytes. We make some assumptions [33,34]: there are only two motes with direct vision, no IEEE 802.15.4 beacon mode is used, and the receiver does not send any ACK after receiving a packet. Fig. 14 shows that the larger the block size, the higher the throughput achieved. Besides, the proposed *OB_HoVe* method provides a throughput which is slightly lower than, the other two techniques, *OB* and *OB_Ho*.

5.8. Simulating attacks

This section analyzes how our protocol responses when a malicious node throws randomly some attack presented in Section 3.6.2. To check the anti-attack capability of our protocol, each experiment is repeated 50 times, 10 data blocks have been sent with messages of 16 byte length and 8 messages each data block check. Table 10 shows the results, 100% of the committed attacks are detected.

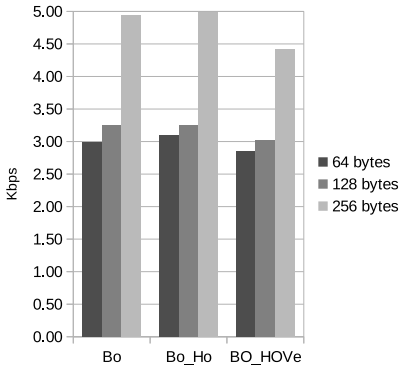

Fig. 14. Throughput.

Table 10
Percentage of detection.

Type of attack	Attacks	Detected
Tamper	355	100%
Replaying	435	100%
Forwarding	420	100%
Forgery	390	100%

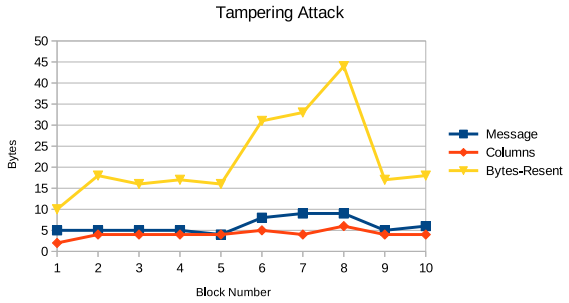

Fig. 15. Tampering Attack Example.

Table 11
Security services.

	OB	TinySec	TeenySec	WaterMarking
Cypher	✗	✓	✓	✗
Integrity	✓	✓	✓	✓
Freshness	✓	✗	✓	✓
Authentication	✓	✓	✓	✓
Side-channel attack	✓	✗	✗	✗

Fig. 15 shows a tampered attack example with 10 blocks. Blue data series represents message number tampered by block. Orange data series represents the amount of tampered bytes by message and yellow data series shows the amount of bytes that need to be resent by a node to the head cluster because of the data block integrity.

6. Comparisons

In this section, the proposed mechanism is evaluated from the security services point of view. In Table 11, these services are shown. The symbol ✗ represents that the service is not implemented, whilst ✓ means that the security service is included.

The *Overlapping Block, (OB)* mechanism is able to provide a simple yet lightweight privacy service by shuffling the positions of the data within the block. Therefore, these methods provide privacy but do not encrypt data. Thus, the processing requirements are lower. Besides, a lower but sufficient level of confidentiality is reached using the *OB_Ho* and *OB_HoVe* methods. The *OB* methods achieve a strong integrity control, because a double integrity control using the *CRC-16* algorithm is included. The freshness in the data is obtained with the sequence number, which prevents from replay attacks. A lightweight authentication service is provided by Bouakkaz et al. [27] algorithm. An additional advantage provided by our methods is the prevention of side-channel attacks, because the sensitive data are sent mixed among the rest of the data. So, it is more difficult for a man-in-the-middle attacker to detect the sensitive data within all the intercepted data. Another important security service is the extreme to extreme security, because the data block can be understood only by sender and receiver.

Table 12
Comparing overhead. Payload: 16 bytes.

	Bytes	% Overhead vs. *base*	% Overhead vs. OB_HoVe
base	16	0%	–
OB	18	11.11%	0%
OB_Ho	18	11.11%	0%
OB_HoVe	18	11.11%	0%
PRW	18	11.11%	0%
FWC-D_bp	18	11.11%	0%
TeenySec	19	15.79%	4.68%
CBC-MAC-4	20	20.00%	8.89%
TinySec	21	23.81%	12.7%
CBC-MAC-6	22	27.27%	16.16%
CBC-MAC-8	24	33.33%	22.22%
FWD-C	34	52.94%	41.83%

6.1. Comparing overhead

This section shows the advantages of our proposed methods in terms of overhead, comparing it with others methods, studied in Section 2. Table 12 shows the overhead introduced by our method and the rest of the studies vs. the *base* version. This *base* version is the standard communication mechanism in *TinyOS* without any additional coding or integrity enhancement. Then, we compare the overhead of other mechanisms with respect to our *OB_HoVe* method. The overhead has been computed assuming messages with 16 bytes of payload.

First column shows the method under comparison. The second column shows the sum of bytes of the payload with additional bytes produced by the security method. The third column represents the overhead percentage of security methods regarding to *base* version. Finally, in the fourth column, we show the overhead percentage of each security method vs. the OB_HoVe proposed mechanism.

These results are graphically shown in Fig. 16. All studied security mechanisms produce overhead respect to *base* version. Our proposed mechanisms generate an overhead of 11.11%, which is one of the lowest along with the watermarking methods. However, watermarking approaches have some drawbacks, which have been highlighted in Section 2.1.4. In this sense, if the underlying WSN does not support distortions in the bits of the data readings to embed the watermark, it produces a large overhead, as it can be seen in the *FWC-D* watermarking method, which has an overhead of 52.94% regarding to the base *TinyOS* version and 41.83% overhead compared to the *OB_HoVe* proposal. On the other hand, if the watermarking is able to modify the precision bits of the data readings, it provides similar overhead to our proposal. This is the case of *PRW* and *FWC-D_bp* methods. Nevertheless, the computational cost to obtain the hash value is much higher than the processing cost of *OB_HoVe*, as shown in Fig. 17.

6.2. Execution time

In this subsection, the execution times of the methods presented in Section 2 and our *OB_HoVe* proposed method are indicated.

Fig. 17 shows the execution time (measured in milliseconds) on *TelosB* nodes of the followings algorithms: *AES-128* using *AES-C* to

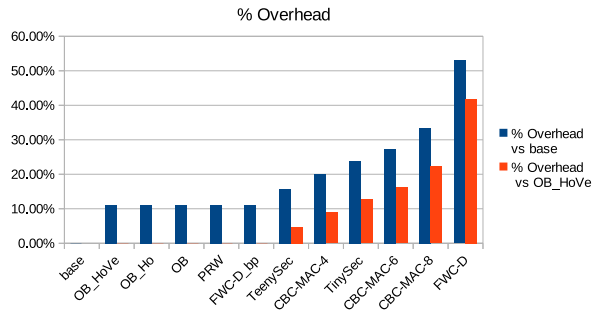

Fig. 16. Comparing Overhead.

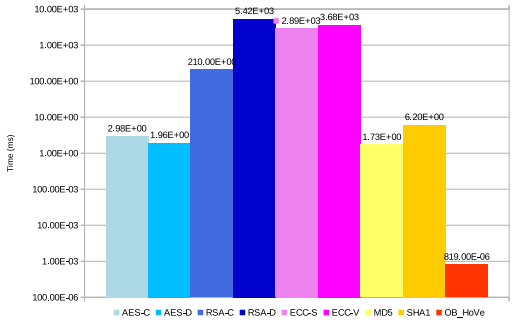

Fig. 17. Execution Time Algorithms.

cypher, and *AES-D* to decypher, both with 16-byte messages; *RSA-1024 bits* using *RSA-C* for encryption and *RSA-D* for encryption. All the measures have been taken from [25]. Finally, ECC is also compared using ECC-S for the cryptographic signature and ECC-V for the verification [26].

However, our *OB_HoVe* method requires only 819 μs to disorder a 128 byte data block, which is much lower than the other algorithms.

6.3. Lightweight privacy

Although the main focus of the proposal is to provide data integrity and a certain degree of security against attacks, the *OB_HoVe* mechanism is able to provide a very lightweight level of privacy to the data without encryption. In this sense, without any prior knowledge of the interchanged data, the proposed method is able to hide the data values within all the rest of the values. If there is no semantic hint applicable to the data, *OB_HoVe* shuffles all the data values of several different variables making original ordering very hard to obtain. Moreover, the CRC values of the rows and columns are sent in the following block. Therefore, there is no clue about the correct ordering of the cells which forms the message. The worst case for an attacker would be to know nothing from the data or the positions on the matrix for the data and the CRC. As we have stated earlier, there is no guess on the data, either semantic or by similarity of the involved values. Therefore, the positions of the cells are unknown and the only way to break the privacy is by guessing the ordering. On the other hand, the best scenario for an attacker is to be able to determine which cells of the matrix include data values and which ones not, because this scenario would reduce the amount of possible data sources. Thus, by brute force attack, if the CRC cell positions are known, for a given $n \times m$ matrix, the total amount of possible rows to be check is provided by the variations without repetitions of $n-1$ values (as one value is the CRC of that row) out of the $n \times m$ values of the shuffled matrix, as described in Eq. (4).

$$V_{(n-1\cdot m-1),(n-1)} = \frac{((n-1)\cdot(m-1))!}{(n\cdot m - m - 2\cdot n + 2)} \tag{4}$$

Each possible row is then CRC-computed and selected if the resulting CRC value is found in the following block. It is expected to find exactly $m-1$ rows that fulfills this requirement. Finally, the ordering of the $m-1$ of the selected rows can be obtained by the permutation of this amount: $V_{rows} = (m-1)!$

For instance, for a 9×9 matrix sending a total amount of 162 bytes, and knowing in advance which cells include CRC values, by brute force, an attacker should check $1.78E + 14$ vectors of 8 values of 16-bits. This huge amount of vectors would require more than 2.8 Petabytes of memory. In a 4-GHz computer running an efficient CRC software implementation would take more than 25 days of computation (without taking into account the memory access latencies). Therefore, considering that most WSN use the received data to take immediate decisions, the exploited data obtained by the attacker is too old and probably, it has been used long time ago by the WSN system. Thus, *OB_HoVe* proposed method provides an enough strong privacy mechanism, if every specific data value cannot be semantically distinguished from the rest of the data values.

7. Discussion

In this section, a comparison of the resource consumption of the proposed scheme and other techniques presented in Section 2 is shown. In the *R-CS* [35] scheme studied to improve the *ARQ Scheme*, no resource consumption measures are provided. However, they proposed to use *CRC64, MD5,* and *Tiger* as one-way hash functions. These methods add a large overhead, in terms of packet length by means of adding 64, 128, and 192 bits, respectively.

TinyMD5 [15] is a modification of *MD5*, proposed to improve the security of *WSNs*. Its evaluation is based on the amount of hash collisions and the energy consumption in a simulated environment. According to the authors, *TinyMD5* shows reductions in communication costs and improvements in the network lifetimes compared to previous schemes. This integrity method increases the overhead in two bytes for each packet compared to our *OB* mechanisms. In *TinyMD5*, the integrity control is centralized on the base station. However, it is worth mentioning that our proposed method carries out the integrity control in a distributed way, in each head cluster.

Regarding *MAC* methods, the *CBC-MAC* is used, for example, in *TinySec* when operating in *TinySec-Auth* authentication mode. This mode increases energy consumption in 3% compared to the original *TinyOS* stack in order to send 24 bytes of data. Also, it increases the overhead in 2 bytes compared to *CRC16*. This architecture is only available in *TinyOS 1.x* and *Mica2* nodes. Using the same measure as [18], *CBC-MAC* code using AES-128 with a 16-byte message consumes 112.2 μJ. Therefore, it entails about 15 ms, which is a much larger time than the one used to compute with CRC16. The hardware of the *CC2420* radio could be used too. However, *CBC-MAC* has not been selected in our proposal because of the flaws described in Section 4.2. Besides *CBC-MAC* has some other weaknesses [18], which prevents us from using it.

With regard to the watermarking methods [21,22,36], it must be mentioned that, only sink nodes or base stations check the integrity of the messages. However, in our scheme, all the head clusters are able to test the integrity and to authenticate the messages. Therefore, messages with errors will not reach the sink node. Thus, it would reduce the network traffic and energy consumption. In Fig. 13 this saving of energy is shown. For example, one packet with 8 errors in different bytes would provoke the retransmission of messages representing about 6.3% of the size of the block but saving 93.7%. On the other hand, most the watermarking methods require redundant bits in the data to embed the watermark. That can be taken as a weakness if the underlying *WSN* does not support those types of distortions in the payload. Besides, in [22], the most significant bits of watermark are chosen. However, selecting only 2, 4, or 8 bits would largely increase the probability of a hash collision. Therefore, it would be necessary to select more bits, but this would increase the overhead.

Our overlapping method is based in a cryptographic principle, named diffusion, which hides the relations between encrypted and plain text. The confidentiality approaches in Sections 2.2.1 and 2.2.2 apply, in addition to diffusion, another cryptographic principle, named confusion. This last principle requires complex math operations. Therefore, these algorithms require more processing time than our overlapping methods, although they provide higher security level. Thus, there is a tradeoff between the security solution chosen and its costs.

8. Conclusions

A lightweight integrity mechanism with overlapping blocks providing multipacket data which is addressed to provide integrity for data messages using the *ITU-T CRC16* has been designed. Three mechanisms have been developed to provide different levels of security: Overlapping Block *(OB)*, Overlapping Block with Horizontal shuffle *(OB_Ho)* and Overlapping Block with complete shuffle *(OB_HoVe)*. According to the experimental tests that have been carried out, the *OB_HoVe* mechanism generates less readable data blocks than the other two proposed mechanisms *(OB* and *OB_Ho)*. However, obtaining that higher level of security implies that the *OB_HoVe* mechanism spends more energy than the others two, about 6μJ as shown in Table 7. On the other hand, the overhead included by *OB, OB_Ho* and *OB_HoVe* is similar, because the main overhead is produced by *CRC-16*, as shown in Table 6. Therefore, we can conclude that the *OB_HoVe* mechanism produces the least readable message, requiring very few additional resources more than the rest of the proposed mechanisms. We have tested this functionality by developing an experiment with a man-in-the-middle module. This module randomly modifies some bytes in the packets. Thus, the receiver is able to detect these errors. Because of the structure of the multipacket, the receiver requests only those columns and rows which are wrong, with a large reduction in the required retransmissions of packets.

Our integrity scheme is distributed, because all the receivers (head cluster and base station) can check the integrity of the data. Therefore, it is unnecessary to wait for the data to arrive at the sink node to check their integrity. This leads to a reduction in the number of exchanged data in the network and, consequently, in the energy consumption as shown in Fig. 13.

Our proposed scheme is interesting, because, on one hand, it applies a lightweight integrity mechanism jointly with overlapping and, on the other hand, the authentication is obtained with a very light method as proposed by [27]. Besides, no software cryptographic algorithm is used, which is an advantage from the point of view of the resource consumption.

CRediT authorship contribution statement

Francisco Alcaraz Velasco: Conceptualization, Methodology, Investigation, Writing – original draft, Software. **Jose Manuel Palomares:** Conceptualization, Validation, Formal analysis, Resources, Writing – review & editing, Supervision. **Joaquin Olivares:** Validation, Resources, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partly supported by the Spanish Ministry of Science, Innovation, and Universities grant RTI2018-098371-B-I00, and, the Advanced Informatics Research Group – GIIA (TIC-252). Funding for open access charge: Universidad de Córdoba / CBUA (Spain).

References

[1] P. Baronti, P. Pillai, et al., Wireless sensor notherswork: A survey on the state of the art and the 802.15.4 and ZigBee standars, Comput. Comun. 30 (7) (2006) 1655–1695, http://dx.doi.org/10.1016/j.comcom.2006.12.020.

[2] A. Belfkih, J. Duvallet, et al., A survey on wireless sensor notherswork databases, Comput. Nothersworks 25 (8) (2019) 4921—4946, http://dx.doi.org/10.1007/s11276-019-02070-y.

[3] J. Horn, A. Koohang, et al., The Internothers of Things: Review and theorothersical framework, Expert Syst. Appl. 133 (2019) 97–108, http://dx.doi.org/10.1016/j.eswa.2019.05.014.

[4] P. Sandeep, W. Wanqing, et al., Medical information security for wearable body sensor nothersworks in smart healthcare, IEEE Consum. Electron. Mag. 8 (5) (2019) 37–41, http://dx.doi.org/10.1109/MCE.2019.2923925.

[5] N.B. Gaathri, G. Thumbur, et al., Efficient and secure pairing-free certificateless aggregate signature scheme for healthcare wireless medical sensor nothersworks, IEEE Internothers Things J. 6 (5) (2019) 9064–9075, http://dx.doi.org/10.1109/JIOT.2019.2927089.

[6] C. Karlof, D. Wagner, et al., Secure routing in wireless sensor nothersworks: Attacks and countermeasures, in: Proceedings of the First IEEE International Workshop on Sensor Notherswork Protocols and Applications, Anchorage (USA), 2003, pp. 113–127, http://dx.doi.org/10.1109/SNPA.2003.1203362.

[7] S. Taisuk, C. Youngho, An enhanced trust mechanism with consensus-based false information filtering algorithm against bad-mouthing attacks and false-praise attacks in WSNs, Electronics 8 (11) (2019) 1359, http://dx.doi.org/10.3390/electronics8111359.

[8] A. Carrasco, F. Alcaraz, et al., Securing a wireless sensor notherswork for human tracking:a review of solutions, Int. J. Commnun. Syst. (27) (2014) 4384—4406, http://dx.doi.org/10.1002/dac.2621.

[9] Fernando León-García, José M. Palomares, Joaquín Olivares, D2R–TED: data–domain reduction model for threshold-based event dothersection in sensor nothersworks, Sensors 18 (2018) 3806, http://dx.doi.org/10.3390/s18113806.

[10] F. León-García, F.J. Rodríguez-Lozano, J. Olivares, J.M. Palomares, Data communication optimization for the evaluation of multivariate conditions in distributed scenarios, IEEE Access 7 (2019) 123473–123489, http://dx.doi.org/10.1109/ACCESS.2019.2936918.

[11] O. Westman, M. Hell, Electromagnothersic side-channel attack on AES using low-end equipment, ECTI Trans. Comput. Inf. Technol. 14 (2) (2020) 139–148, http://dx.doi.org/10.37936/ecti-cit.2020142.239925.

[12] Y. Wang, G. Attebury, et al., A survey of security issues in wireless sensor nothersworks, IEEE Commun. Surv. Tutor. 8 (2) (2006) 2–23, http://dx.doi.org/10.1109/COMST.2006.315852.

[13] T.C. Maxino, P.J Koopman, et al., The effectiveness of checksums for embedded control nothersworks, Ieee Trans. Dependable Secur. Comput. 6 (1) (2007) 59–72, http://dx.doi.org/10.1109/TDSC.2007.70216.

[14] R.L Rivest, RFC1321: The MD5 message digest algorithm, Tech. rep., RFC Editor, USA, 1992.

[15] B. Kyoungsoo, L. Yunjeong, et al., An energy-efficient secure scheme in wireless sensor nothersworks, J. Sensors (2016) 11, http://dx.doi.org/10.1155/2016/1321079.

[16] C. Karlof, N. Sastry, et al., TinySec: A link layer security architecture for wireless sensor nothersworks, in: Proceedings of the 2Nd International Conference on Embedded Nothersworked Sensor Systems, New York (USA), 2004, pp. 162–175.

[17] L. Knudsen, D. Wagner, On the structure of Skipjack, Discrotherse Appl. Math. (1) (2001) 103–116, http://dx.doi.org/10.1016/S0166-218X(00/00347-4.

[18] L. Jongdeog, K. Kapitanova, et al., The price of security in wireless sensor nothersworks, Comput. Nothersworks 54 (17) (2010) 2967–2978, http://dx.doi.org/10.1016/j.comnothers.2010.05.011.

[19] G. Pereira, R. Alves, et al., Performance evaluation of cryptographic algorithms over IoT platforms and operating systems, Secur. Commun. Nothersworks (2046735) (2017) 16.

[20] M. Dener, O. Faruk, TeenySec: a new data link layer security protocol for WSN, Secur. Commun. Nothersworks (18) (2016) 5882–5891, http://dx.doi.org/10.1002/sec.1743.

[21] K. Ibrahim, J. Hussam, A lightweight data integrity scheme for sensor nothersworks, Sensors 11 (4) (2011) 4118–4136, http://dx.doi.org/10.3390/s110404118.

[22] G. Zhang, L. Kou, et al., A new digital watermarking mothershod for data integrity protection in the perception layer of IoT, Secur. Commun. Nothersworks 2017 (2017) 12, http://dx.doi.org/10.1155/2017/3126010.

[23] A. Menezes, C. van Oorschot, S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996, p. 794.

[24] M. Cazorla, S. Gourgeon, et al., Survey and benchmark of lightweight block ciphers for MSP430 16-bit microcontroller, Secur. Commun. Nothersworks (18) (2015) 3564–3579, http://dx.doi.org/10.1002/sec.1281.

[25] U. Gulen, A. Alkhodary, et al., Implementing RSA for wireless sensor nodes, Sensors (13) (2019) 15, http://dx.doi.org/10.3390/s19132864.

[26] A. Liu, P. Kampanakis, P. Ning, TinyECC: elliptic curve cryptography for sensor nothersworks, in: International Conference on Information Processing in Sensor Nothersworks (Ipsn 2008), 2008, pp. 245–256, http://dx.doi.org/10.1109/IPSN.2008.47.

[27] F. Bouakkaz, M. Omar, et al., Lightweight sharing scheme for data integrity protection in WSN, Wirel. Pers. Commun. (1) (2016) 211–226, http://dx.doi.org/10.1007/s11277-016-3261-5.
[28] Standard, IEEE, 802.15.4, http://www.ieee802.org/15/pub/TG4.html, accessed: 2019-10-30.
[29] Instruments datasheothers chipcon smartrf cc2420, Available from: http://www.ti.com/lit/ds/symlink/cc2420.pdf, accessed: 2019-10-30.
[30] A. Diaz, P. Sanchez, Simulation of attacks for security in wireless sensor notherswork, Sensors 16 (11) (2016) 1–27, http://dx.doi.org/10.3390/s16111932.
[31] J. Polastre, R. Szewczyk, et al., Telos: enabling ultra-low power wireless research, in: IPSN 2005. Fourth International Symposium on Information Processing in Sensor Nothersworks, Boise (USA), April 2005, pp. 364–369, http://dx.doi.org/10.1109/IPSN.2005.1440950.
[32] Philip Levis, David Gay, TinyOS Programming, Cambridge University Press, 2009.
[33] L. Benoît, D. Mil, et al., Throughput and delay analysis of unslotted IEEE 802.15.4, J. Nothersworks 1 (1) (2006) 20–28, URL http://dx.doi.org/1854/7653.
[34] T. Sun, L. Chen, et al., Measuring effective capacity of IEEE 802.15.4 beaconless mode, in: IEEE Wireless Communications and Nothersworking Conference, Vol. 1, 2006, pp. 493–498, http://dx.doi.org/10.1109/WCNC.2006.1683513.
[35] Q. Zhang, J. Xiao, et al., Improve security of wireless sensor nothersworks through reluctant checksum, Int. J. Distributed Sens. Nothersworks 13 (9) (2017) 7, http://dx.doi.org/10.1177/1550147717731041.
[36] S. Xingming, S. Jianwei, Digital watermarking mothershod for data integrity protection in wireless sensor nothersworks, Int. J. Secur. Appl. (4) (2013) 407–416.

Francisco Alcaraz was born in Córdoba, Spain, in 1977. He received a B.Sc. in Computer Science in 1998 from the Universidad de Córdoba, Spain. M.Sc. in 2003 from Universidad de Málaga, Spain. M.Sc in Computer and Network Engineering in 2011 from the Universidad de Sevilla, Spain. M.Sc. in Communication, Networks and Content Management in 2016 from the Universidad UNED, Spain. M.Sc. in Intelligent Systems in 2016 from the Universidad de Córdoba, Spain. His research interests are in the field of security of wireless sensor networks and embedded systems.

Jose M. Palomares was born in Motril, Spain, in 1975. He received a Ph.D., M.Sc., and B.Sc. degrees in Computer Engineering from the University of Granada, Granada, Spain in 2011, 1998 and 1996, respectively. Since 2000, he has been working as a Lecturer, Assistant Professor, and, currently, Associate Professor at the Universidad de Córdoba, Córdoba, Spain. He is co-founder of the Advanced Informatics Research Group, Universidad de Córdoba, Córdoba, Spain. He has research interests in Image and Video Processing, Real-Time Systems, Wireless Sensor Networks, and Computer Architecture.

Joaquín Olivares is an Associate Professor at Electronic and Computer Engineering Department at the Universidad de Córdoba, Córdoba, Spain, since 2001. He received the B.S. and M.S. degrees in Computer Sciences in 1997, and 1999, respectively, and the M.S. degree in Electronics Engineering in 2003, all from the Universidad de Granada, Spain. He received the Ph.D. degree in 2008 at the Universidad de Córdoba, Spain. His research interests focus on embedded systems, wireless sensor networks, computer vision, and high-performance computing.