SAD-SJ: a Self-Adaptive Decentralized solution against Selective Jamming attack in Wireless Sensor Networks

Marco Tiloca, Domenico De Guglielmo, Gianluca Dini and Giuseppe Anastasi
Department of Information Engineering, University of Pisa
Largo Lucio Lazzarino 2, 56122, Pisa, Italy
{marco.tiloca, domenico.deguglielmo, gianluca.dini, giuseppe.anastasi}@iet.unipi.it

Abstract

Wireless Sensor Networks (WSNs) are currently used in many application scenarios, including industrial applications and factory automation. In such scenarios, Time Division Multiple Access (TDMA) is typically used for data communication among sensor nodes. However, TDMA-based WSNs are particularly prone to Selective Jamming attack, a specific form of Denial of Service attack aimed at severely thwarting network reliability. In this paper, we present SAD-SJ, a self-adaptive and decentralized MAC-layer solution against selective jamming in TDMA-based WSNs. SAD-SJ does not need a central entity, requires sensor nodes to rely only on local information, and allows them to join and leave the network without hindering other nodes activity. We show that SAD-SJ introduces a limited overhead, in terms of computation, communication and energy consumption.

1. Introduction

Wireless Sensor Networks (WSNs) are currently used in a large number of application domains, including industrial applications such as factory automation, distributed and process control, robotic networks, real-time monitoring of machinery health, detection of liquid/gas leakage, radiation check, and so on [1][2]. Usually, *energy efficiency* is the major constraint in the design of WSN-based systems [3], since sensor nodes are typically powered by batteries, with limited power budget. However, in industrial application scenarios, additional requirements need to be considered, such as *scalability*, *reliability*, *timeliness,* and *security* [2]. In such scenarios, Time Division Multiple Access (TDMA) is typically used for communication among sensor nodes. In TDMA, time is divided into a sequence of periodic superframes, each one composed of a fixed number of transmission slots that are pre-assigned to sensor nodes. Each sensor node is active only during its own slot, and sleeps for the rest of the time, thus saving energy. Therefore, TDMA provides guaranteed bandwidth, high energy efficiency, absence of collisions (i.e. high reliability), as well as low and predictable latency.

On the other side, in TDMA-based sensor network, communication is particularly vulnerable to *jamming attack*, a specific form of *Denial of Service* (*DoS*) attack aimed at thwarting network availability [4]. Basically, jamming consists in interfering in network operational frequencies, thus corrupting packets transmitted by legitimate users. Jamming is considered a severe issue in WSN communication [5][6].

TDMA is prone to a particularly insidious form of jamming attack, namely *Selective Jamming* (*SJ*) [7][8]. This kind of attack aims at disturbing communication among sensor nodes according to specific criteria and objectives. For instance, an adversary could be interested in jamming only the transmission of certain packet types, or packets sent in one specific TDMA slot of the superframe, or transmissions from a specific sensor node. Also, selective jamming could severely compromise specific traffic flows. In comparison with traditional wide-band jamming, selective jamming is more difficult to be detected, due to the reduced adversary exposure. In the following, we focus on one kind of selective jamming attack, according to which the adversary aims at disrupting communication of one specific sensor node.

Selective jamming attack has been thoroughly investigated in literature [7][8]. Typical defenses proposed against it are physical layer solutions, and rely on spread-spectrum communication among sensor nodes [4][5][6][7][8][9][10]. In [11], the authors propose a multichannel MAC protocol aimed at enhancing transmission efficiency while resisting wireless interference and jamming. Furthermore, a MAC layer countermeasure against selective jamming in IEEE 802.15.4 sensor networks [12] using the GTS mechanism has been proposed in [13]. GTS is basically a form of TDMA communication where slots in different superframes are allocated to sensor nodes by the coordinator node. Hence, it is extremely vulnerable to selective jamming attacks. The authors have proposed a centralized solution where the slot allocation pattern is randomly changed at each superframe by the coordinator node.

However, the presence of a coordinator node is the main limitation of centralized sensor networks. Such a node represents a single *point of failure*, i.e. if it fails, the

network goes out of service. Also, in a dynamic environment, where the number of active sensor nodes may change over time, sensor nodes have to explicitly associate/de-associate with the coordinator node upon joining/leaving the network. This results in additional overhead and energy consumption on sensor nodes.

In this paper, we propose *SAD-SJ*, a *self-adaptive* and *decentralized* solution against SJ attacks in sensor networks using a generic TDMA scheme for communication. SAD-SJ has a number of strengths. First, it neutralizes the selective jamming attack, forcing the adversary to carry out a random attack. Assuming there are N transmission slots in the superframe, this results in the attack effectiveness reduced to $1/N$. In addition, no centralized entity is required to contrast selective jamming, and network nodes rely only on local information. Finally, SAD-SJ is self-adaptive, i.e. nodes can join and leave the network at any time, without hindering other nodes activities. We also show that a join operation in our scheme is always completed in, at most, one superframe (while leave operations are instantaneous). Finally, while SAD-SJ has been conceived for TDMA-based sensor networks, it can also be easily extended to slotted communication schemes such as solutions based on de-synchronization algorithms [14][15][16][17].

We evaluated the performance and overhead of SAD-SJ through a network model based on a *Discrete Time Markov Chain* (*DTMC*). We show that the introduced overhead, in terms of computation, communication, and energy consumption, is limited.

The rest of this paper is organized as follows. Section 2 describes the SJ attack we refer to, while in Section 3 we present and discuss our decentralized solution SAD-SJ. Section 4 describes how new nodes join the network in the presence of SAD-SJ. In Section 5, we discuss the impact of our solution on network performance and energy consumption, and provide an analysis of the join overhead. Finally, in Section 6 we draw our conclusive remarks.

2. SJ attack in TDMA sensor networks

In the following, we assume that time is divided into periodic *superframes* of equal duration. A superframe T consists of N equally sized transmission slots, each one of which can be used to transmit one packet. In addition, we assume that there are $U \leq N$ sensor nodes, and each sensor node u has been permanently assigned one slot s_u, in the superframe. Hence, node u can use slot s_u in all superframes.

We consider an adversary whose objective consists in disrupting communications of a specific target node, namely u. In order to do that, she performs a form of selective jamming by maliciously transmitting during slot s_u assigned to sensor node u, thus interfering with its network activities.

In a traditional TDMA-based sensor network, such an attack is very easy to be performed. It is sufficient to observe a single superframe T in order to determine the slot s_u associated to sensor node u. Then, since u is supposed to access slot s_u in all superframes, it is possible to thwart its communications with an effectiveness of 100%. Besides, since the adversary jams one single slot per superframe, she can turn off the radio during all other slots. From the adversary standpoint, this makes such an attack also extremely efficient in terms of energy consumption. In addition, it exposes the adversary for a very limited amount of time, thus making it difficult to detect her presence.

3. Our solution against SJ

In this section, we describe SAD-SJ, our decentralized and self adaptive solution to the SJ attack. Our approach is based on the following two independent components.

1. **Random Slot Reallocation**. Starting from an initial slot allocation pattern, at each superframe T, a new (pseudo) random slot allocation pattern S is generated. Specifically, S is computed by sensor nodes in an autonomous way, i.e. relying only on local information. In such a way, the adversary is no more able to jam her victim communications, unless by interfering with a randomly selected slot. This reduces the attack effectiveness to *1/N*.
2. **Network Dynamicity Management.** Network dynamicity is specifically addressed, so that sensor nodes can join and leave the network, at any time, without interfering with other nodes activities. New incoming nodes operate the join process in a fully autonomous way, i.e. without requiring any central entity.

3.1. Initialization

Before deployment, all sensor nodes are provided with the following information.

i) A secret cryptographic symmetric key K, namely *permutation key*, whose size discourages brute force attack;
ii) an unsigned integer number z_0 whose maximum possible value is Z, i.e. $z_0 \in \{0, \ldots, Z\}$;
iii) the number of slots in each superframe, namely N.

In addition, each node u locally maintains an unsigned integer z whose maximum possible value is Z, i.e. $z \in \{0,\ldots,Z\}$, and a bit vector v_u of N elements, initialized as $v_u[i] = 0, \forall\ i = \{0,1,\ldots,N\text{-}1\}$.

Once the initial slot allocation has been completed, each sensor node is associated to one TDMA slot. Then, each node u, associated to the i-th slot in the superframe, performs the following steps:

(i) updates its local vector v_u so that $v_u[i] = 1$;
(ii) initializes z as $z = z_0$.

Then, each sensor node protects itself from the SJ attack, according to the procedure described in Section 3.3. In order to do that, sensor nodes rely on the slot permutation procedure described in Section 3.2.

3.2. Slot permutation

In order to produce a pseudo-random re-allocation of TDMA slots, each sensor node u needs to produce a random permutation of v_u elements. Such a process is performed according to the following steps. Given a vector v of N elements, producing a pseudo-random permutation of its elements requires to i) generate N pseudo-random numbers; and ii) perform N swap operations, which displays $\mathcal{O}(N)$ computational complexity.

In order to produce permutations of vector v, sensor nodes must rely on a secure pseudo-random number generator [18]. Provided that they all rely on the same initial *seed* quantity, all sensor nodes are able to generate the same unpredictable sequence of values over time, and thus remain synchronized with one another during the slot permutation procedure. Also, this must be possible while relying only on local information, i.e. without any information exchange.

This can be achieved by selecting a common initial *seed* value, and then applying a *one-way function* to a well known sequence of values. Examples of suitable one-way functions include cryptographic hash functions, such as *SHA-1* [19], or block ciphers, such as *AES-128* [20]. Although such ad-hoc methods have not been proven to be cryptographically secure, they appear sufficient for most applications [18]. The *rand()* function shown below provides an example of pseudo-random number generation based on the approach described above. In the following, we assume the presence of a secure and efficient symmetric cipher, that relies on a secret cryptographic key.

```
int rand() {
   int random_value = encrypt(z, K);
   z = (z + 1) % (Z + 1);
   return random_value;
}
```

We recall that K is the permutation key shared by all sensor nodes, z is an integer value stored by all sensor nodes, and Z is the maximum possible value of z. Basically, the *rand()* function returns a pseudo-random value computed as the output of a symmetric cipher. The encryption process takes the permutation key K and the current value of z as input. Before returning the new random value, z is incremented by 1. Note that, upon reaching the maximum possible value Z, z is assigned to 0, i.e. $z = 0$. Since all nodes initialize z with the same value z_0, it follows that the i-th generated number, namely r_i, is always the same for all nodes.

Besides, it is worth emphasizing that, with reference to commercially-available WSN platforms such as *Tmote Sky* [21], the actual encryption process can be performed through hardware, with negligible processing overhead in terms of both delay and energy consumption [22].

Thanks to the random number generation described above, the actual permutation process can be performed by each node, according to the following *permute()* function. Its behavior is equivalent to the *random_shuffle()* function provided by the C++ Standard Template Library [23].

```
void permute(int v[]) {
   for (int i = 0; i < N; i++)
   swap(v[i], v[rand() % N]);
}
```

We recall that v is the vector stored by each node, and composed of N elements. The loop in the *permute()* function considers one element of v per step, and swaps it with a randomly chosen element of the same vector. As a special case, an element can be possibly swapped with itself. Also, it is possible that the vector v remains unchanged after the permutation process has been completed.

Of course, once the *rand()* function has been invoked Z times, i.e. $z = z_0$, every node starts producing a sequence of random numbers identical to the previous one. This can make it easier for the adversary to perform cryptanalysis, so making the generation of random numbers, and thus the TDMA slot permutation process, predictable. In order to solve this issue, it is sufficient that each node updates the permutation key K used by the *rand()* function, once $z=z_0$. For instance, given an encryption function $E_y(x)$ that encrypts a quantity x by means of a key y, the new permutation key K^+ can be obtained by using the old permutation key K to encrypt itself, i.e. $K^+ \leftarrow E_K(K)$. We discourage renewing z, since it would not result in the generation of a totally different pseudo-random sequence. Also, it would require to achieve consensus among sensor nodes in a distributed way. Accomplishing such a task may not be easy, and is likely to result in a not negligible additional overhead and energy consumption.

3.3. Protection against jamming

Let T_0 be the first superframe after the slot allocation process has been completed. As described in Section 3.1, each node u is associated to the i-th slot of T_0. Also, its permutation vector v_u includes $v_u[i] = 1$, and $v_u[j] = 0$, $\forall$ $j \neq i$.

Hereafter, at the end of every superframe T_m, each node u locally determines the slot s_u to refer to during the next superframe, namely T_{m+1}, according to the following procedure.

1. Node u produces a permutation v^*_u of its local vector v_u, by invoking the *permute()* function described in Section 3.2, providing it with vector v_u. Then,

2. u replaces v_u with the permutated vector computed at step 1, i.e. $v_u \leftarrow v^*_u$. Then,
3. u determines i such that $v_u[i] = 1$. Finally,
4. u selects the i-th slot as its own slot s_u of superframe T_{m+1}.

It is worth noting that vectors v of different nodes always contain different configurations. Therefore, given any two distinct nodes u and w, results of their permutation process, namely v^*_u and v^*_w, are likely to be different vectors. However, this is not an issue from the jamming protection standpoint. In fact, every node u is interested only in producing its local permutation v^*_u, and then locating the i-th entry of v_u such that $v_u[i] = 1$, in order to determine the slot s_u. Also, at superframe T_0, every node has been exclusively associated to an available TDMA slot.

Finally, swaps performed by the *permute()* function do not take into account values of v_u elements, but consider only their position within the vector. Thus, for each pair of nodes u and w, if $v_u[i] = 1$ and $v_w[j] = 1$, then $i \neq j$. As a consequence, at each superframe, every node exclusively accesses a TDMA slot, and no collisions occur.

3.4. Discussion

According to SAD-SJ, sensor nodes randomly generate a new slot allocation pattern at each superframe, efficiently and autonomously. Thus, at the end of each superframe T_m, every node u is able to correctly determine the slot s_u to be used during superframe T_{m+1}. In addition, SAD-SJ guarantees that slot s_u is exclusively assigned to node u, i.e. there are no collisions with other nodes. From the adversary standpoint, her victim, say node u, accesses the medium during a slot which generally changes on a per-superframe basis. Thus, in order to carry on the attack and disrupt u activities, the adversary is forced to pick one of the N slot at random, hoping it is assigned to u in the current superframe. This drastically reduces the effectiveness of the SJ attack to *1/N*.

4. Node leave and join

In this section, we describe how sensor nodes leave and join the network. First, we describe the case when one or more sensor nodes leave, or are forced to leave, the network. SAD-SJ is not affected at all by nodes leaving the network, as they just release their own slot, which becomes available. The remaining sensor nodes continue with their normal operations. However, in order to assure and maintain network security, it is necessary to provide a new permutation key K to the remaining sensor nodes, by excluding and thus logically evicting the leaving ones. This problem is commonly known as *rekeying*, and has been widely discussed in literature [24][25][26][27]. Although we included it in our solution, this issue is beyond the scope of this paper.

Now, we describe the *join procedure,* i.e. the process performed by a sensor node to join the network. For the sake of simplicity, we assume that, at each superframe, at most one sensor node attempts to join the network. The case of multiple-node joins is left for further study. We show that, under such hypothesis, the join process is extremely efficient and brief, as any sensor node is able to complete the join procedure in, at most, one superframe.

In order to assure that new nodes can correctly join the network, each of the currently present nodes has to transmit additional information during its own slot. Specifically, at superframe T_m, each node u broadcasts also one additional information, that is the value assumed by z at the beginning of T_m. Such an information needs to be authenticated, in order to assure that it has been broadcast by legitimate network nodes. So doing, new nodes are able to synchronize themselves with the ongoing slot permutation process.

Let us assume that a node u wants to join the network, and at least one slot is available, i.e. the number of active nodes is less than N. Node u is equipped with the shared permutation key K (provided to any sensor node before the physical deployment). Node u initializes its local vector v_u as $v_u[i] = 0$, $\forall\, i = \{0,1,...,N-1\}$. Then, the join procedure takes place as follows.

1. Node u listens to transmissions from other nodes during the current superframe, namely T_j, retrieves the current value of z, namely z_j, and verifies its authenticity. Then,
2. u locates the first free slot in superframe T_j, i.e. the i-th slot in the superframe. Finally,
3. u initializes its local z as $z=z_j$, and updates its local vector v_u so that $v_u[i] = 1$.

As to step 1, in the worst case, only one sensor node is active and is using the last slot in the superframe. Hence, u is forced to listen to the entire superframe. However, in general, node u needs to listen to only a number of slots, up until it finds one assigned slot (to get the z value) and one available slot (to be used for subsequent operations). Hereafter, node u is able to correctly protect itself against the selective jamming attack, by using the countermeasure described in Section 3.3.

5. Overhead analysis

In this section, we perform an analysis of the overhead caused by SAD-SJ. First, we derive the additional energy consumption due to our solution when the network is in steady state conditions. Then, we perform an analysis aimed at estimating both the duration of the *join procedure* and the average energy consumed by a sensor node to join the network.

5.1. Overhead in steady-state conditions

As shown in Section 3, SAD-SJ requires only trivial vector element swaps, and random number generation through encryption. Also, the additional transmission of z requires such information to be authenticated. However, both encryption and authentication operations

can be efficiently performed by hardware components, provided by most of the available sensor platforms [21], such as the *CC2420* chipset [28]. Thus, the overhead introduced by SAD-SJ is almost negligible from a computational standpoint. Instead, the actual security overhead is mostly due to the transmission of additional security information, that typically includes a specific authentication field, namely *Message Integrity Code* (*MIC*) [22]. To fix ideas, let us assume to represent z on 4 bytes. Also, we consider different sizes of the *MIC* field, namely 4, 8, and 16 bytes, as prescribed by the *IEEE 802.15.4* standard [12]. Of course, the larger the *MIC* field, the more data authenticity is guaranteed in case of an exhaustive brute force attack. Also, we denote with S the total additional number of bytes sent by every node during its own slot (i.e. $S = z$ + *MIC* size). Finally, we consider the average transmission power consumption of the *CC2420* chipset [28], i.e. P_{TX} = 31.32 mW, and a data transmission rate R = 250 Kbit/s [12].

z size	*MIC* size	S	E_{TX}
4 Bytes	4 Bytes	8 Bytes	8.017 μJ
4 Bytes	8 Bytes	12 Bytes	12.026 μJ
4 Bytes	16 Bytes	20 Bytes	20.045 μJ

Table 1. Additional energy consumption.

Table 1 shows the additional energy per superframe consumed by each sensor node, namely E_{TX}, computed as $E_{TX} = P_{TX} \cdot ((S \cdot 8) / R)$. As it can be observed, such an overhead depends on the total amount of additional bytes sent, and results to be moderate and affordable.

5.2. Join overhead

In this section, we characterize the duration of the *join procedure*, i.e. the time taken by a node to join the network. Also, we measure the additional energy consumption due to the *join procedure*. To this end, we derive a Discrete-Time Markov-Chain (DTMC) model for the *join procedure*.

We define T_j as the superframe at which a node u starts the join procedure, and S_j as the slot allocation pattern at superframe T_j. Also, we indicate with N_A and N_F, $(N_A + N_F) = N$, the amount of assigned and available slots in T_j, respectively. Finally, we indicate with D_{slot} the duration of a slot, and with P_{RX} the power consumption of the radio while in receive mode.

As described in the previous section, node u has to perform essentially two tasks: i) receive the actual value of z from one of the N_A permanent nodes, and ii) locate the first available slot in the superframe. Once the two tasks have been completed, u switches off its radio, and waits for the beginning of the next superframe. As it can be observed, the time required by a node to complete the *join procedure* depends on the specific slot allocation pattern S_j. Specifically, the duration of the join procedure can range from only 2 slots to N slots. The former case occurs when the first slot of superframe T_j is assigned and the second slot is available, or vice versa. The latter case occurs when there is only one node in the network and it accesses the last slot in T_j. During superframe T_j, each slot s_i, $i \in \{1, \dots, N\}$, has a probability equal to $\frac{N_A}{N}$ and $\frac{N_F}{N}$ to be either assigned or available, respectively.

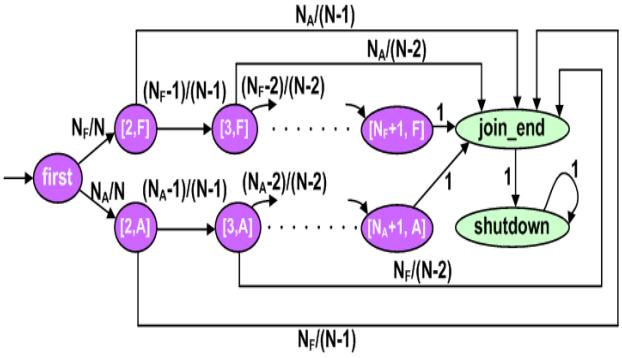

Figure 1. DTMC for the join procedure.

The system is observed at the beginning of each slot s_i, $i \in \{1, \dots, N\}$, and the state of the system is represented by the state of the joining node at the beginning of each time slot. Initially, u is in state *first*, indicating that it is observing the first slot of superframe T_j. Instead, it is in state *join_end* when it has retrieved all the necessary information to join the network, and in state *shutdown* if it has switched off its radio. Also, u is in state $[i_f, F]$, $i_f \in \{2, \dots, N_F + 1\}$, if it examines slot s_{i_f} and it has found only available slots before s_{i_f}. Instead, it is in state $[i_a, A]$, $i_a \in \{2, \dots, N_A + 1\}$ when it observes slot s_{i_a} and it has found no available slots before s_{i_a}.

In the analysis, we indicate the set of possible states of the chain as Ω. Since there are $(N + 3)$ different states for the joining node u, $|\Omega| = (N + 3)$. Figure 1 reports the graph representation of the DTMC.

Now, we derive the transition probabilities for the Markov chain, i.e. the probability P_{XY} of passing from a state X to another state Y, $\forall X, Y \in \Omega$. Initially, u observes the first slot in the superframe, i.e. it is in state *first*. Then, it moves either to state $[2, F]$ or $[2, A]$, in case it finds the first slot to be available or assigned, respectively. The probabilities of such transitions to occur are equal to $\frac{N_F}{N}$ and $\frac{N_A}{N}$, respectively.

Now, let us assume that u is in state $[2, F]$. It moves either to state *join_end* if the second slot is assigned, or to state $[3, F]$ if the second slot is found available. The transition $[2, F] \rightarrow join\_end$ occurs with a probability equal to $\frac{N_A}{N-1}$. In fact, there are still $(N - 1)$ slots to be visited in the superframe, and no assigned slots have been found yet. Instead, the transition $[2, F] \rightarrow [3, F]$ has a probability equal to $\frac{N_F - 1}{N-1}$ to occur, since there are $(N_F - 1)$ available slots out of $(N - 1)$ remaining slots.

Now, we examine the transitions that can occur when u is in state $[2, A]$. There are two possible transitions: i)

transition $[2,A] \rightarrow [3,A]$ that occurs when the second slot is assigned, and ii) transition $[2,A] \rightarrow join\_end$ which occurs when the second slot is available. Since there are $(N-1)$ slots to be visited and (N_A-1) assigned slots, transition i) occurs with probability $\frac{N_A-1}{N-1}$. Instead, the probability of transition ii) to occur is equal to $\frac{N_F}{N-1}$ since there are only N_F available slots among the remaining $(N-1)$ slots.

The abovementioned considerations can be easily extended to all states $X = [i_f, F]$, $i_f \in \{2, \dots, N_F\}$, and $X = [i_a, A]$, $i_a \in \{2, \dots, N_A\}$. More specifically, the probability to pass from state $[i_f, F]$ to state $[i_f+1, F]$, i.e. $P\{[i_f, F], [i_f+1, F]\}$, is equal to:

$$P\{[i_f, F], [i_f+1, F]\} = \frac{N_F - i_f + 1}{N - i_f + 1}$$

Similarly, the transition from state $[i_a, A]$ to state $[i_a+1, A]$, i.e. $P\{[i_a, A], [i_a+1, A]\}$, can be calculated as follows:

$$P\{[i_a, A], [i_a+1, A]\} = \frac{N_a - i_a + 1}{N - i_a + 1}$$

Instead, the transition probabilities from states $[i_f, F]$ and $[i_a, A]$ to state *join_end* can be calculated as:

$$P\{[i_f, F], join\_end\} = \frac{N_A}{N - i_f + 1}$$

$$P\{[i_a, A], join\_end\} = \frac{N_F}{N - i_a + 1}$$

In case u has visited $N_F(N_A)$ consecutive available (assigned) slots, i.e. it is in state $[N_F+1, F]([N_A+1, A])$, it moves to state *join_end* with a probability equal to 1, since there are no more available (assigned) slots in the superframe. Finally, once u has reached the state *join_end*, it moves to state *shutdown* with probability 1, and remains there since it is an absorbing state.

Once we have derived the transition probability P_{XY}, $\forall X, Y \in \Omega$, we can obtain the transition probability matrix P of the Markov chain. To this end, we sort the states of the Markov chain so that *first* and *join_end* are the first and last state in the sequence, respectively. Let v_0 denote the initial probability vector, and v_k the probability vector after k slots, $k = \{1,2, \dots, N\}$.

Without loss of generality, we can assume $v_0 = [1,0, \dots, 0]$. Hence, $v_k = v_0 \cdot P^k$. The probability that the *join procedure* has a duration of exactly k slots, i.e. $P_{join}(k)$, corresponds to the probability of being in state *join_end* at step k, i.e. $v_k[|\Omega|]$.

Now, we can derive the average energy spent by node u to join the network. We denote with $N_{max} = \max(N_A, N_F) + 1$ the maximum number of slots required to join the network. Also, we define $E_{slot} = (P_{RX} \cdot D_{slot})$ as the energy consumed by a joining node to observe a single time slot. Thus, if the *join procedure* has a duration of exactly k slots, the energy consumed by a joining node can be calculated as $k \cdot E_{slot}$. Instead, the average energy spent for the join procedure, i.e. E_{avg}, is equal to $E_{avg} = \sum_{k=2}^{N_{max}} k \cdot E_{slot} \cdot P_{join}(k)$. In fact, we must take into account all possible durations of the *join procedure*. Hence, the sum considers any possible value $k \in \{2, \dots, N_{max}\}$ of the joining time, which occurs with probability $P_{join}(k)$. Inside the sum, the product between k, $P_{join}(k)$, and E_{slot} is performed.

5.3. Results

In this section, we show the results obtained from the analytical model we previously derived. In order to validate our analytical results, we relied also on simulation, and implemented SAD-SJ using the *ns-2* simulation tool [29]. We considered a single-hop network, where sensor nodes are located at a fixed distance (10 m) from the sink node. The transmission range was set to 15 m (according to the settings reported in [30]), while the carrier sensing range was set to 30 m, as in [31]. Unless stated otherwise, other parameter values are as shown in Table 2. For every experiment, we performed ten independent replications, each one of which consists of 1000 join procedures. Results shown below are averaged over all replications. We also derived confidence intervals by using the independent replication method and 95% confidence level.

Parameter	Value
Slot duration (D_{slot})	7.4 ms
Power Consumption in RX mode (P_{RX})	35.46 mW
Number of slots (N)	10, 30, 50

Table 2. Parameters used in our analysis.

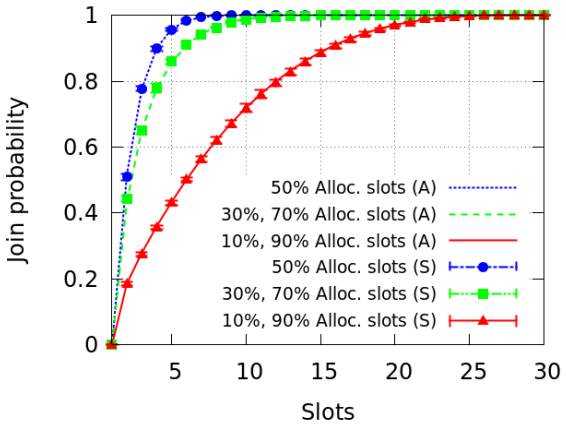

Figure 2. Joining time distribution (N = 30).

Figure 2 shows the probability distribution of the join duration, when the superframe is composed of 30 slots, for different percentages of allocated slots. It is derived both from analysis and simulation, whose results almost

overlap. Hence, the model is validated. As it can be observed, the fraction of allocated slots significantly affects the duration of the joining operation. Specifically, the join duration decreases as the percentage of allocated slots tends to 50%.

This is because, in such a case, the number of allocated and available slots is approximately the same. Thus, the joining node has a high probability to quickly locate both an available and an assigned slot. Instead, if there are few available or assigned slots in the superframe (i.e. the percentage of assigned slots is very high or very low), then the joining node has in general to visit a high number of slots before it can complete the join procedure. Also, note that, from a probabilistic standpoint, there are no differences between the case in which there are x% allocated slots or x% available slots in the superframe.

Allocated slots (%)	***N* (# of slots)**		
	10	**30**	**50**
10, 90	10	19	22
50	4	5	6
30, 70	6	8	8

Table 3. 95% joining time (# of slots).

Table 3 reports the 95-th percentile of the joining time distribution for different values of N and percentages of assigned slots, i.e. the number of slots required to complete a join with a probability of, at least, 0.95.

As it can be observed, the time required to join the network increases with the number of slots in the superframe. However, even with $N = 50$, the joining time remains low. Also, at most one superframe is required to join the network.

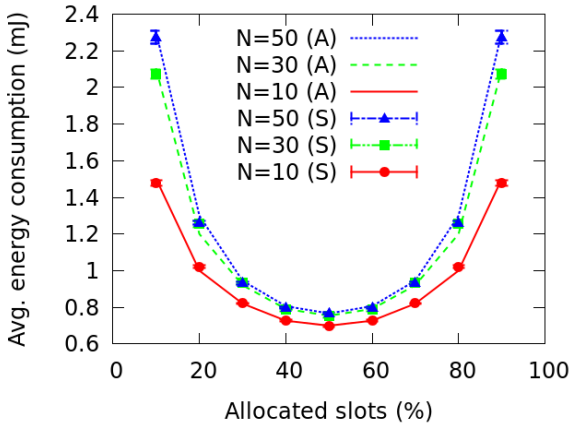

Figure 3. Join average energy consumption.

Figure 3 reports the average energy consumed to join the network, for N = {10,30,50}, with different percentages of assigned slots. As above, analytical and simulation results almost overlap. As it can be observed, in all considered scenarios, the average energy consumption displays a parabolic trend. Also, as expected, energy consumption displays a minimum in case the percentage of assigned slots is equal to 50%, while it increases if the percentage of assigned (available) slots is very low (high). Considerations made for the convergence time are valid also for energy consumption.

6. Conclusion

In this paper, we have presented SAD-SJ, a novel decentralized solution against selective jamming attack in TDMA-based WSNs. Our solution neutralizes the selective jamming attack, by forcing the adversary to perform a random attack, thus reducing its effectiveness to $1/N$, where N is the total number of slots in the superframe. We have shown that SAD-SJ is self-adaptive, as it allows nodes to join and leave the network at any time and without affecting security of other nodes. Finally, SAD-SJ displays a negligible impact on network performance, and results in an additional energy consumption which is limited and affordable. Future work will extend our solution against selective jamming, in order to address multiple nodes attempting to join the network at the same time.

7. Acknowledgment

This work has been supported by the EU FP7 Integrated Project PLANET (Grant agreement n. FP7-257649); the TENACE PRIN Project (n. 20103P34XC) funded by the Italian Ministry of Education, University and Research; and the Regione Toscana POR CReO PITAGORA.

References

[1] A. Willig, "Recent and Emerging Topics in Wireless Industrial Communications: a Selection", *IEEE Transactions on Industrial Informatics*, Vol. 4, N. 2, pp. 102-124, May 2008.

[2] R. Zurawski, "Networked Embedded Systems: An Overview", Chapter 1 in *Networked Embedded Systems* (R. Zurawski, Editor), pp. 1.11-1.16, CRC Press, 2009.

[3] G. Anastasi, M. Conti, M. Di Francesco and A. Passarella, "Energy Conservation in Wireless Sensor Networks: a Survey", *Ad Hoc Networks*, Vol. 7, N. 3, pp. 537-568, May 2009.

[4] D.R. Raymond and S.F. Midkiff, "Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses", *IEEE Pervasive Computing*, Vol. 7, N. 1, pp. 74-81, 2008.

[5] L. Lazos, S. Liu and M. Krunz, "Mitigating control-channel jamming attacks in multi-channel ad hoc networks", *Proceedings of the second ACM conference on Wireless network security (WiSec '09)*, ACM, New York, NY, USA, pp. 169-180, 2009.

[6] W. Xu, K. Ma, W. Trappe and Y. Zhang, "Jamming sensor networks: attack and defense strategies", *IEEE Network*, Vol. 20, N. 3, p. 41-47, 2006.

[7] A. Proano and L. Lazos, "Selective Jamming Attacks in Wireless Networks", *Proceedings of the 2010 IEEE*

International Conference on Communications, pp. 1-6, 2010.

[8] R. Sokullu, I. Korkmaz and O. Dagdeviren, "GTS Attack: An IEEE 802.15.4 MAC Layer Attack in Wireless Sensor Networks", *International Journal On Advances in Internet Technologies*, Vol. 2, N. 1, pp. 104-114, 2009.

[9] D.R. Raymond and S.F. Midkiff, "Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses", *IEEE Pervasive Computing*, Vol. 7, N. 1, pp. 74-81, 2008.

[10] R. Pickholtz, D. Schilling and L. Milstein, "Theory of Spread Spectrum Communications - A Tutorial", *IEEE Transactions on Communications*, Vol. 30, N. 5, pp. 855-884, 1982.

[11] L. Tang, Y. Sun, O. Gurewitz and D. B. Johnson, "EM-MAC: a dynamic multichannel energy-efficient MAC protocol for wireless sensor networks", *Proceedings of the Twelfth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '11)*, pp. 23:1-23:11, 2011.

[12] Institute of Electrical and Electronics Engineers, Inc. New York, *IEEE Std. 802.15.4-2006, IEEE Standard for Information technology – Telecommunications and information exchange between systems - Local and metropolitan area networks - Specic requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*, September 2006.

[13] R. Daidone, G. Dini and M. Tiloca, "On preventing GTS-based Denial of Service in IEEE 802.15.4", *Poster and Demo Proceedings of the 9th European Conference on Wireless Sensor Networks (EWSN 2012)*, pp. 69-70, 2012.

[14] J. Degesys, I. Rose, A. Patel and R. Nagpal, "DESYNC: Self-Organizing Desynchronization and TDMA on Wireless Sensor Networks", *Proocedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN 2007)*, pp. 11-20, Cambridge, USA, 2007.

[15] R. Pagliari, Y. Hong and A. Scaglione, "Bio-Inspired Algorithms for Decentralized Round-Robin and Proportional Fair Scheduling", *IEEE Journal on Selected Areas in Communications (J-SAC)*, Vol. 28, N. 4, 2010.

[16] H. Kang and J. Wong, "A Localized Multi-Hop Desynchronization Algorithm for Wireless Sensor Networks", *Proceedings of IEEE INFOCOM 2009*, pp. 2906-2910, Rio de Janeiro, Brazil, 2009.

[17] A. Motskin, T. Roughgarden, P. Skraba and L. Guibas, "Lightweight Coloring and Desynchronization for Networks", *Proceedings of IEEE INFOCOM 2009*, pp. 2383-2391, Rio de Janeiro, Brazil, 2009.

[18] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, USA, 2001.

[19] National Institute of Standards and Technology (NIST), *Secure Hash Standard*, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, 2008, http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf

[20] National Institute of Standards and Technology, *Federal Information Processing Standards Publication 197, Specification for the ADVANCED ENCRYPTION STANDARD (AES)*, November 2001.

[21] Moteiv Corporation, *Tmote iv Low Power Wireless Sensor Module*, Moteiv Corporation, San Francisco, CA, USA, November 2006, http://www.cs.jhu.edu/~cliang4/public/datasheets/tmote-sky-datasheet.pdf

[22] R. Daidone, G. Dini and M. Tiloca, "On experimentally evaluating the impact of security on IEEE 802.15.4 networks", *Proceedings of the 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS 2011)*, pp. 20-25, June 2011.

[23] B. Stroustrup, *The C++ Programming Language (3rd ed.)*, Addison-Wesley, Boston, USA, 2000.

[24] C. K. Wong, M. Gouda and S. S. Lam, "Secure group communications using keygraphs", *IEEE/ACM Transactions on Networking*, Vol. 8, N. 1, pp. 16-30, February 2000.

[25] G. Dini and M. Tiloca,"HISS: a HIghly Scalable Scheme for group rekeying", *The Computer Journal*, pp. 1-18, 2012, in press.

[26] S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication", *ACM Computing Surveys*, Vol. 35, N. 3, pp. 309-329, September 2003.

[27] G. Dini and I. M. Savino, "LARK: A Lightweight Authenticated ReKeying Scheme for Clustered Wireless Sensor Networks", *ACM Transactions on Embedded Computing Systems*, Vol. 10, N. 4, pp. 41:1-41:35, 2011.

[28] Texas Instruments, *CC2420 2.4 GHz IEEE 802.15.4 / ZigBee ready RF Transceiver*, 2012, http://focus.ti.com/lit/ds/symlink/cc2420.pdf

[29] Network Simulator Ns2, http://www.isu.edu/nsnam/ns

[30] J. Zheng and M. J. Lee, "A comprehensive performance study of IEEE 802.15.4", *IEEE Press Book*, 2004.

[31] G. Anastasi, M. Conti, M. Di Francesco and A. Passarella, "A Comprehensive Analysis of the MAC Unreliability Problem in 802.15.4 Wireless Sensor Networks", *IEEE Transactions on Industrial Informatics*, Vol.7, N.1, pp. 537-568, February 2011.