Secure Communication Protocol for Wireless Sensor Networks*

D. Rossi M. Omaña D. Giaffreda C. Metra
ARCES – DEIS, University of Bologna, Italy
{d.rossi, martin.omana, daniele.giaffreda2, cecilia.metra}@unibo.it

Abstract

We propose a new communication protocol for wireless sensor networks, allowing to make them secure with respect to malicious attacks. Compared to standard secure protocols (e.g., the IEEE 802.15.4 and the ZigBee), the one we propose allows to increase security significantly, at negligible impact on node complexity. A possible hardware scheme to implement our protocol is also proposed.

1. Introduction

Modern microprocessors ability to process large amounts of data and implement advanced digital signal processing is increasingly enabling the successful deployment of advanced control systems based on remote sensing and measuring [1]. This is opening a widespread range of new opportunities to control the environment, wildlife habitats, complex industrial plants, aerospace vehicle platforms, etc [1].

Wireless personal area networks (WPANs), defined by the IEEE 802.15.4 standard [2], are widely adopted within the abovementioned applications, because of their low complexity and cost [3]. Particularly, the ZigBee protocol [4] is becoming a *de facto* standard for WPANs.

One important issue for WPANs is their security, in terms of confidentiality, integrity, authenticity, availability [5, 6]. In fact, differently from a wired network, where the information transmitted among nodes is confined within a physical medium (a wire), in a wireless network the information is exchanged among nodes through electromagnetic waves, that are broadcasted to the atmosphere. This makes them prone to various kinds of attacks that may try to violate the security of the network [7, 6] like, for instance, the *Denial of Service* (*DoS*), the *Man-in-the-middle* (MITM), and the c*opy and repeat* (denoted also as *replay*) attacks [7]. As a consequence, especially in case of WPANs employed within control systems for safety critical applications (e.g., chemical or nuclear industrial plants, aerospace vehicle platforms, etc) it is of utmost importance to implement countermeasures to guarantee their security [6].

*Work partially supported by Becar – Beghelli (Italy).

In order to provide a WPAN with an adequate security level, it is essential that the nodes exchanging information are able to provide assurance on their identity, and that the exchanged information is authentic and integral [7, 6].

The IEEE 802.15.4 standard and the ZigBee Alliance protocol provide the communication with some level of authenticity and integrity. Particularly, they are able to prevent/limit the abovementioned attacks on the messages transmitted from the transmitter (master) node (Tx) to receiver (slave) nodes (Rx) [2]. Reversely, they do not provide any protection to the acknowledgment (ACK) message [5], that is sent back by the Rx in order to confirm to the Tx that the message has been successfully received. The lack of protection for the ACK is a serious bug [5], possibly harming the security of the whole network. In fact, this makes a protocol prone to MITM-like attacks. Moreover, in the IEEE 802.15.4 and the ZigBee protocols, the message freshness against possible *replay* attacks is guaranteed by a frame counter only. This makes the WPAN vulnerable to DoS attacks, for instance exploiting the overflow of a frame counter [5].

Based on these considerations, in this paper we propose a new, secure communication protocol for WPANs, that is able to guarantee message integrity, authenticity and freshness for both sender messages and acknowledgment messages. Furthermore, our protocol prevents the *copy and repeat* attack by a different, and more secure approach, compared to the IEEE 802.15.4 and ZigBee protocols.

The rest of this paper is organized as follows. In Section 2, we discuss some important security problems of standard WPAN protocols. In Section 3, we describe our proposed protocol. In Section 4, we present a possible hardware implementation of our protocol. Finally, Section 5 concludes the paper.

2. Security Problems of Standard Protocols

As introduced above, in standard IEEE 802.15.4 and ZigBee Alliance protocols, when the Rx receives the message (MSG), it verifies its validity (integrity, authenticity and freshness). If the message is valid, then the Rx sends an unprotected ACK to the Tx,

978-1-4244-9556-6/10/$26.00 ©2010 IEEE

which considers the communication successfully concluded upon its receipt, with no check upon the ACK authenticity (i.e., its coming from the expected Rx node) and freshness (i.e., its being a new ACK, and not an old ACK copied and repeated by a non authorized node).
The lack of authentication and freshness verification on the ACK could seriously threaten the security of the whole network. For instance, let us consider the simple case of an attacker trying to avoid that a message arrives to the Rx, thus starting a MITM attack. It can first simply send an interference noise to the Rx at the same time as the Tx, thus preventing the Rx from properly receiving the MSG sent by Tx. Afterwards, the attacker can send a fake ACK to the Tx. This way, since the ACK is not protected, the Tx can be fooled that Rx successfully received the MSG.
Another limit of standard protocols is that they verify the MSG freshness by checking a sequence number provided by a simple counter. They are consequently prone to DoS-like attacks [8]. In fact, as shown in [8], such an attack can be carried out by assembling a fake message that is compliant with the protocol format and by setting its sequence number equal to the maximum counting value. This way, the counter of the node receiving such a fake message will overflow. This will make the counters of the Tx and Rx no longer synchronous, so that the Rx will discard all following messages from Tx.

3. Proposed Secure Protocol

We propose a protocol to overcome the security problems described in the previous section.
As significant example, we consider the case of a master/slave network with one master node, acting also as network manager, and several slave nodes. However, our protocol can be applied to any kind of WPAN by means of straightforward modifications.
Similarly to the standard IEEE 802.15.4 and ZigBee protocols, our protocol guarantees authentication and integrity of the MSG by means of a Message Authentication Code (MAC) [2], that is generated by encrypting the message in clear text by an AES algorithm [9]. Moreover, differently from the standard IEEE 802.15.4 and ZigBee protocols, our protocol: i) guarantees the authenticity and freshness of the ACK, thus avoiding MITM attacks; ii) provides a new mechanism to guarantee the freshness of the MSG, that is able to protect the WPAN with respect to DoS-like attacks.
To guarantee the authenticity of the ACK, we propose to generate a MAC for the ACK, by encrypting the ACK in clear text by an AES algorithm [9].
To guarantee the freshness of the ACK and the MSG, as described in more details later in this section, our protocol embeds a rolling code sequence [10] (#seq) on both the ACK and MSG, and provides a original mechanism to synchronize the Tx and Rx, that makes the WPAN immune to DoS-like attacks.
The general structure of the derived MSG or ACK is schematically shown in Fig. 1.

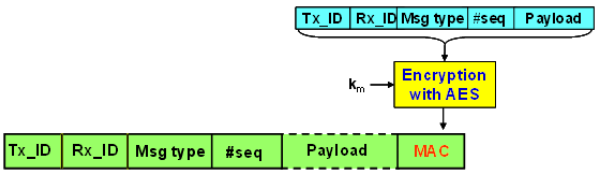

Fig. 1. Structure of a MAC or ACK of our protocol.

The fields Tx_ID and Rx_ID contain the identification codes (IDs) of the transmitter Tx and receiver Rx, respectively. The Payload is the useful part of the message, containing data, commands or ACKs. The field Msg_type indicates the type of message (e.g., command, data, ACK, synchronization, etc.) included in the payload field. The field #seq contains the current sequence number of the rolling code of the node (Tx/Rx) sending the message. Finally, the MAC of the MSG or ACK is generated by encrypting the first part of the message (i.e., Tx_ID, Rx_ID, Msg_type, #seq and Payload) by the AES algorithm and a secret key k_m.
As we can see from Fig.1, the useful message (i.e., the whole message, but for the MAC) is sent in clear text. This allows to identify the Rx of the message and to verify the correctness of #seq without any decryption, thus reducing power consumption and the impact on communication latency. Of course, this approach can not be used if the message carries critical data, while it does not give rise to any security flow if the message consists of a simple command, as it is usually the case in WPANs.
Let us describe in details how our protocol guarantees the freshness of both the AKC and MSG.
As for the adopted rolling code sequence, it can be a pseudo-random sequence, for instance generated by a *Linear Feedback Shift Register* [11].
The freshness of the ACK and MSG is verified when the rolling code sequences (#seq) in the master (Tx) and in the slave (Rx) nodes are synchronized. This is achieved in the following way.
Let's suppose, as an example, that the WPAN is composed by one Tx node (N1), and three Rx nodes (N2, N3 and N4), as schematically represented in Fig. 2. Tx must generate/memorize a different #seq for any

Rx node. Thus, in a network of *n* nodes, Tx must generate *n-1* different sequences (seq(1,i), i=2..n), while each Rx must generate only one #seq (the respective seq(1,i) sequence), as represented in Fig. 2.

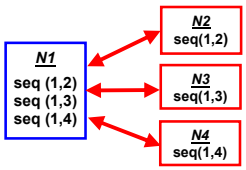

Fig. 2. Sequences of the rolling code memorized on a master node (N1) and slave nodes (N2, N3, N4).

In our protocol, the synchronism between the #seq of Tx and Rx is guaranteed by the ACK. When a Rx (Ni, with i=2, 3, 4) receives a message (MSG) from Tx, it verifies the authenticity and freshness of the received MSG and, in case these verifications are successful, it accepts the MSG, updates its seq(1,i) and sends back an ACK to Tx (N1). When Tx receives the ACK, it also updates its seq(1,i) that is specific to the Rx with which it was communicating. Therefore, upon conclusion of a successful communication, Tx and Rx are synchronized. If Rx (Tx) receives a MSG (ACK) with a #seq different from the expected one (e.g., due to an attack), Rx (Tx) discards the MSG (ACK) without updating its #seq. This way, the synchronization of the #seq of Tx and Rx is still guaranteed, and the Rx is ready to accept a possible new (valid) message from Tx.

Additionally, our protocol provides a retransmission procedure that avoids the loss of synchronism also when Tx does not receive any ACK from Rx (e.g., if due to excessive noise in the channel, the MSG sent by Tx does not reach Rx, or the ACK sent by Rx does not reach Tx). After sending a MSG, Tx triggers a timer and waits for the arrival of the ACK for a proper time interval t_w. If after t_w Tx has not yet received the ACK, it sends again the MSG to Rx and triggers again the timer. The t_w is chosen large enough to allow the MSG to reach Rx and be processed by it, to permit Rx to elaborate the ACK, and the ACK to reach node Tx.

Tx repeats this retransmission procedure till it receives the ACK from Rx, or till the maximum number of retransmissions (n_{max}) allowed by our protocol is reached. If after n_{max} retransmissions Tx has not received any ACK, it labels Rx as "problematic" node.

Retransmissions are managed by node Rx as follows. When Rx receives a MSG with a #seq equal to its previous #seq, then it compares the whole MGS with the last MSG received from Tx. If they match, then Rx recognizes it as a retransmitted message and sends again the ACK to Tx without increasing its #seq.

Finally, as for the Rx nodes that have been labeled as "problematic", our protocol provides the following resynchronization procedure between Tx and Rx.

Initially, the master Tx starts sending to Rx a synchronization MSG (which is built as shown in Fig. 1, but without any payload) with its current $\#seq_{TX}$. When Rx receives this kind of MSG, it makes its own sequence $\#seq_{RX}$ equal to $\#seq_{TX}$. Then, Rx sends and ACK to Tx with $\#seq_{RX}$ and increases its sequence by 1 (i.e., $\#seq_{RX}+1$). When Tx receives the ACK, it also increases its sequence by 1 (i.e., $\#seq_{TX}+1$) and sends an ACK with the updated sequence to Rx. Finally, when Rx receives the ACK, it verifies that $\#seq_{TX}+1 = \#seq_{RX}+1$. If they are the same, then Rx assumes that Tx is an authorized node of the network, thus accepting the new sequence. Otherwise, Rx keeps its old sequence number.

4. Implementation and Verification

In order to verify the effectiveness of our protocol, we designed the Tx and Rx blocks implementing our protocol in Verilog and we synthesized them with Altera Quartus II [11].

As an example, we considered the case of a single master (Tx) node and three slave (Rx) nodes (Fig. 2). Moreover, we considered a field of 2 bits for the Tx ID, for the Rx ID and for the message type, while we considered a field of 6 bits for the Payload. Additionally, and without loosing generality, we have considered a rolling code implemented by a 3 bit counter. Finally, we used a 128 bits AES module (i.e., input message and secret key of 128 bits) to generate the message MAC. Thus, before the encryption of the message (to generate its MAC), the message is expanded to 128 bits by adding 0s.

Our implementation of the Tx node is schematically shown in Fig. 3. As for the Rx node, its structure is very similar and is not shown for space limitation.

Tx is divided in two functional blocks: i) Tx-part, which elaborates the MSGs to be sent to the Rx; ii) Rx-part, which controls the ACK messages received from Rx.

As for the Tx-part, it receives as input: i) the ID of the Rx (ID_Rx), ii) the message type (MsgT), iii) the Payload, iv) the current rolling code sequence number (My_RC) of the Tx generated by the 3 bit counter *RC_Gen*, v) the IDs of problematic Rx nodes (ID_PN), which are loaded during the initialization phase into a proper table within the block *Rx_Status*.

Before starting a communication with a Rx, *Rx_Status* verifies that Rx is not a problematic node. If Rx is not problematic, *Rx_Status* sets its output Set_act_com to 1, thus indicating to the *ACL* block (which keeps track

of the Rx nodes with which Tx has active communications) to add the node Rx to the list of nodes with active communications. In addition, when Set_act_com=1, *Elab_MSG* is enabled and starts elaborating the MSG to be sent (as in Fig. 1). Then, *Elab_MSG* sets Msg_Ready to 1, to indicate that the message given to its output (Out MSG), and loaded into the output register *reg 3*, is ready to be sent.

Simultaneously, when Msg_Ready = 1: i) the timer *Start_Tw* (which accounts for the maximum time that Tx waits for the arrival of the ACK from Rx) is enabled, and ii) the counter *Count_Ret* (which keeps track of the number of retransmissions) is incremented by one.

As for the Rx-part, its *Verify_ACK* block is enabled when timeout=1, which takes place when *Start_Tw* reaches its maximum count.

The block *Verify_ACK* verifies the correctness of the Rx ID and the rolling code sequence (#seq) of the received ACK. If the Rx ID is correct and the #seq of the ACK is the same as the expected one (i.e., equal to My_RC), then *Verify_ACK* sets the signal Load_ACK to 1. When Load_ACK=1, *Verify_MAC* is enabled. This block regenerates the MAC from the clear text of the input ACK and compares it with the MAC received in the ACK. If both the received and the regenerated MACs are equal, then the input ACK is considered valid and *Verify_MAC* sets the signal Valid_ACK to 1. This indicates that the communication with Rx has been accomplished successfully. Then, Rx is removed from the list of nodes with active communications in *ACL*, and *RC_Gen* increases My_RC by 1.

Instead, if in the input ACK the IDs of the Rx or Tx are not valid, or the #seq is not equal to My_RC, then signal Load_ACK = 0 (Load_ACK# = 1) and the input ACK is discarded before analyzing its MAC. The input ACK is also discarded if its MAC is not correct. In this case it is Valid_ACK = 0 (Valid_ACK# = 1).

When Load_ACK# or Valid_ACK# are set to 1, the signal ReTx is set to 1 and the Out_MSG is retransmitted to Rx. Then, *Start_Tw* is restarted, and the number of retransmissions is incremented by 1 in *Count_Ret*. If *Count_Ret* reaches n_{max}, it sets the signal Update_PN to 1, which labels the Rx as "problematic".

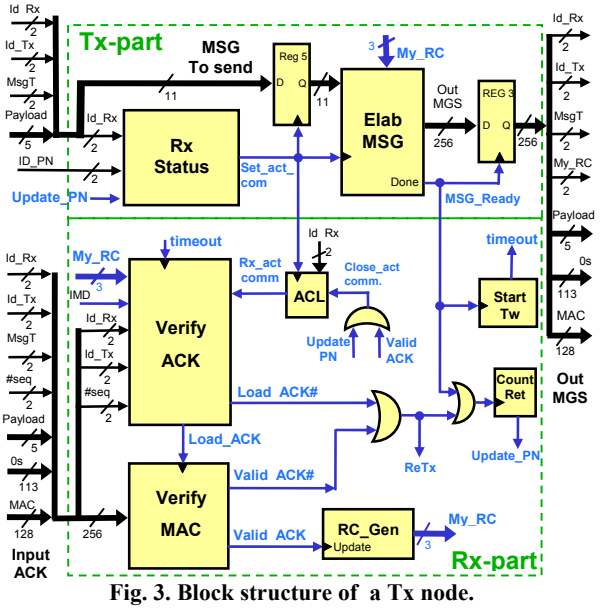

Fig. 3. Block structure of a Tx node.

6. Conclusions

We have proposed a new communication protocol for wireless sensor networks, allowing to make them secure with respect their most common attacks. Compared to the standard IEEE 802.15.4 and ZigBee protocols, our protocol allows to increase security significantly, at negligible impact on node complexity. Finally, we also presented a possible hardware scheme to implement our protocol.

References

[1] A. Tiwari, P. Ballal, F.L. Lewis, "Energy-Efficient Wireless Sensor Network Design and Implementation for Condition-Based Maintenance", *ACM Trans. on Sensor Networks*, Vol. 3, No. 1, March 2007, pp. 1-22.

[2] Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks, *IEEE Standard*, 802.15.4-2003, May 2003.

[3] C. Sinem, "ZigBee/IEEE 802.15.4 Summary", 2004.

[4] Zigbee Alliance. http://www.zigbee.org.

[5] J. Zheng, J. Li, M. J. Lee, M. Anshel, "A Lighweight Encryption and Authentication Scheme for Wireless Sensor Networks", *Int. J. Security and Networks*, Vol. 1, No 3/4, 2006.

[6] L. Zhou, Z. J. Haas, "Securing Ad Hoc Networks", *IEEE Network*, pp. 24-30, November/December 1999.

[7] D. Dzung, M. Naedele, T. P. Von Hoff, M. Crevatin, "Security for Industrial Communication Systems", *Proc. of the IEEE*, Vol. 93, No. 6, pp. 1152-1177, June 2005.

[8] N. Sastry, D. Wagner, "Security Considerations for IEEE 802.15.4 Networks", in *Proc. of ACM Workshop on Wireless Security*, October 2004.

[9] Joan Daemen, Vincent Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard." *Springer*, 2002. ISBN 3-540-42580-2.

[10] Secure Rolling Code Algorithm for Wireless Link, Application Note AVR411.

[11] Altera Quartus II - Web Edition, https://www.altera.com