Improving Key Negotiation in Transitory Master Key Schemes for Wireless Sensor Networks

Improving Key Negotiation in Transitory Master Key Schemes for Wireless Sensor Networks

Cesare Celozzi, Filippo Gandino, and Maurizio Rebaudengo

Polytechnic of Turin, Department of Automation and Information Technology,
Corso Duca degli Abruzzi, 24, 10129 Turin, Italy
{cesare.celozzi,filippo.gandino,maurizio.rebaudengo}@polito.it

Abstract. In recent years, wireless sensor networks have been adopted in various areas of daily life, and this exposes the network data and hardware to a number of security threats. Many key management schemes have been proposed to secure the communications among nodes, for instance the popular LEAP+ protocol. This paper proposes an enhanced variant of the LEAP+ protocol that decreases the key setup time through the reduction of the number of packets exchanged. This improves the security of communications. The results obtained by network simulation after extensive testing are compared to the corresponding data derived from the LEAP+ protocol to quantify the improvements.

Key words: key management, wireless sensor networks, transitory master key.

1 Introduction

Wireless sensor networks (WSNs) have obtained worldwide attention in recent years due to the diffusion of Micro-Electro-Mechanical Systems technology which has led to the manufacture of smart sensors. These sensors are smaller and more affordable than the older generation sensors and can measure and collect information from the environment, transmit this data through wireless communication links and process them in order to take decisions. However, these sensor nodes have limited computing resources and can only perform complex tasks in large regions if organized in an interlinked network.

Nowadays this pervasive technology is exploited in various applications ranging from infrastructure monitoring [1] to HVAC (heating, ventilation, and air conditioning) for buildings [2]. WSNs have also been applied to military purposes [3] due to the low costs and high scalability. In each of these contexts communications security is crucial. In particular, WSNs must be protected from threats that could compromise the integrity and confidentiality of the data or alter the behavior of the nodes. Since WSNs are often deployed in unsafe or hostile areas they are exposed to various security threats like eavesdropping, hardware tampering or injection of malicious requests. Therefore, in order to protect the integrity, confidentiality and reliability of WSNs an effective security scheme is required.

The key aspect of the security in WSNs is the protection of the communications between pairs of sensor nodes. In principle, the network links can be protected through asymmetric cryptography techniques which allow the key distribution to be managed efficiently. However, given the low computational resources of the sensor nodes and the limited power supply [4], [5], symmetric cryptography has been largely exploited in the majority of recent security schemes. Symmetric cryptography can be used to satisfy the main security requirements, such as authenticity and confidentiality. The adoption of a symmetric encryption scheme implies that each pair of nodes of the WSNs shares a secret key. The negotiation of these cryptographic keys (*key management* [6][7]) is independent of the employed encryption method and heavily affects the security, the computational load and power consumption of the WSN.

Various approaches based on symmetric cryptography have been proposed in the context of key management [8], [9], [10]. In transitory master key techniques all nodes share a master key which is deleted after a certain amount of time (*key setup time*). This is estimated to be the time required by the WSN to negotiate a pairwise key for each pair of nodes. The security assumption is that the key setup time is shorter than the time required by an attacker to extract the master key from a compromised node.

Among the above mentioned approaches, LEAP/LEAP+ protocol and its variants [11], [9], [12] have emerged as effective transitory master key protocols for pairwise key negotiation in static WSNs which allow node addition. LEAP+ protocol relies on the difficulty in accessing the memory of a deployed node containing the master key before its deletion which occurs few seconds after the deployment. The secrecy of the master key is crucial for the security of links since all the pairwise keys are derived from a pseudo-random function indexed by the master key and applied to the IDs of the node. Therefore, a shorter key setup time implies lower probability of key theft and higher security of the communication links.

This paper proposes a modified version of the pairwise key negotiation protocol of the LEAP+ to reduce the key setup time. This goal is achieved through a set of variations of the pairwise key negotiation handshake which decreases the number of packets exchanged in the wireless channel reducing the number of collisions and thus the handshake time. The reduction of the handshake time allows the adoption of a shorter key setup time, keeping the percentage of negotiated pairwise keys constant. The data extracted from network simulations of the LEAP+ protocol and of the proposed enhanced variant have been illustrated and compared in order to quantify the benefits of the modifications.

The remainder of the paper is organized as follows: in Section 2 the LEAP+ protocol together with an overview of the main security issues is described. Section 3 presents the proposed modification of the handshake. Finally, in Section 4, the proposed approach is evaluated and compared with the original protocol, and in Section 5 some conclusions are drawn.

2 Overview of LEAP+ protocol

The LEAP+ protocol is based on a transitory master key technique and on the assumption that a newly deployed node cannot be compromised within a short period of time (denoted by T_{MIN}). This is the time required for neighbor discovery and pairwise key negotiation. Therefore, T_{MIN} represents the maximum amount of time available to an attacker to access and copy the memory of a sensor node. The security scheme presented in [9] proposes the adoption of four kinds of key to manage different levels of communication among the nodes (including the Base Station). This work is focused on the pairwise key negotiation which is the most crucial security aspect of the LEAP+ protocol. In order to increase the security level, the pairwise keys of each pair of nodes are negotiated after the deployment, exploiting the transitory master key secrecy. In this way each pair of nodes will have a different shared secret and the compromise of one pairwise key will not affect the security of the other links of the network.

The pairwise key negotiation procedure is composed of 4 phases, as shown in Fig. 1, where:

- $a \longrightarrow *$: node a broadcasts a packet;
- $a \longrightarrow b$: node a unicasts a packet to node b;
- $\{m\}_K$: message m cyphered with key K;
- $MAC(m)_K$: Message Authentication Code of the message m indexed by the key K;
- $a|b|c$: concatenation of a, b, c.

Before the deployment of the network an offline setup procedure (*Phase 0*) is carried out. During this phase the central controller generates and loads the same transitory master key on each node of the network. From the transitory master key each node derives its own private master key $K_u = f_{K_{IN}}(u)$, where $f(\cdot)$ is a pseudo-random function indexed by the key K_{IN}. At the time of deployment each node starts a timer which measures the lifetime of the master key. When the timer elapses the master key is deleted from the memory. In this way the node will no longer be able to start a handshake procedure for the negotiation of a pairwise key since it is no longer capable of verifying the authenticity of the $ACK1$ answer. However a node which is no longer in possession of the master key can still answer to any $HELLO$ message received from other nodes. Therefore, this mechanism allows the addition of new nodes to a network that has already completed the deployment phase.

After key initialization the nodes are ready for deployment. When a node is activated and deployed it starts to exchange messages with its neighbors to negotiate the pairwise keys. In order to start the handshake, a generic node u periodically broadcasts a packet called $HELLO$. This packet contains the identification code ID_u of the sender. Through this packet the node communicates its presence to the neighbors (*Phase 1*). The frequency of the $HELLO$ packets ($1/T_{HELLO}$) has great impact on the performance of the handshake. In fact, the transmission of a high number of messages increases the probability that

Phase 0, Key initialization

Phase 1, Send HELLO:
$u \longrightarrow * : ID_u$

Phase 2, Send ACK1:
$v \longrightarrow u : ID_v, MAC(ID_u|ID_v)_{K_v}$

Phase 3, Send ACK2:
$u \longrightarrow v : ID_u, MAC(ID_v)_{K_v}$

Fig. 1. Handshake for pairwise key negotiation in LEAP+ protocol.

every neighbor will receive the $HELLO$ message but also increases the number of collisions on the wireless channel. Therefore, sending $HELLO$ packets with high frequency degrades the overall performance of the system. The choice of a proper T_{HELLO} must be made taking into account the average node degree of the network.

A generic node v which receives the $HELLO$ message will reply with an acknowledgment message $ACK1$ (*Phase 2*). This message is unicast to the sender of the $HELLO$ message and contains the ID_v and a MAC indexed by the private master key K_v. In order to avoid collisions the $ACK1$ packets are sent after the *backoff* time which is a random time extracted from a uniform distribution with range $(0, T_{BACKOFF})$. At the same time the node starts a timer that will elapse after an interval of time during which the node waits for an answers ($ACK2$) from the $HELLO$ sender (node u). If the node does not receive the $ACK2$ message after the timer elapses, it retransmits the $ACK1$ message. This retransmission is scheduled in the interval of time ($T_{BACKOFF}$+1s, 2·$T_{BACKOFF}$+1s).

When the node u receives back the $ACK1$ message it verifies the integrity and authenticity of the message computing the MAC and comparing it to the one attached to the received message. In positive cases it generates and stores the pairwise key and sends a response called $ACK2$ (*Phase 3*). The $ACK2$ contains the ID_u of the $HELLO$ sender and the MAC for authentication. When the node v receives the $ACK2$ and verifies the integrity and authenticity of the message the handshake is completed and both nodes share the same pairwise key for further secure communications.

2.1 Security issue

From a security point of view the main weak point of LEAP+ protocol is that the compromise of the transitory master key during the deployment phase may disrupt the security of the whole network. In fact, an attacker in possession of the master key may decipher eavesdropped traffic and even fabricate new nodes able to initiate the handshake for pairwise key negotiation. The threshold T_{MIN} represents the interval of time during which it can be assumed that it is not physically possible to compromise the memory of a node. However, the

experiment realized in 2005 by [10] showed that it is possible to obtain a copy of the memory of a node in tens of seconds. This study also showed that the key setup time may last minutes depending on the average node degree and on the number of messages exchanged. Since T_{MIN} must not be longer than the time estimated by [10] (which future technologies will lower), the reduction of T_{MIN} is a critical aspect for ensuring the security of the key management scheme. This work focuses on this security issue and proposes a variation of the LEAP+ that dramatically lowers the value of T_{MIN} required by LEAP+ for networks with same average node degree.

3 Proposed approach

From the analysis presented in the previous section it can be noticed that in specific cases, especially those with high average node degree, the LEAP+ protocol does not allow the negotiation of all the keys actually available in the system because T_{MIN} is too short. The major cause of this behavior is the high number of collisions generated by the large quantity of messages exchanged in a small time interval during the negotiation phase. A possible solution for this problem is the adoption of T_{MIN} intervals with longer duration. However, this solution increases the probability of compromising a node that is still in possession of the master key, thus allowing an attacker to break all network communications. Conversely, the violation of a node which is in possession of the sole private master key only allows the violation of the communications that involve the compromised node. Therefore, to increase the security of the network the time interval T_{MIN} should be minimized. To achieve this goal the handshake ($HELLO \longrightarrow ACK1 \longrightarrow ACK2$) should be as efficient as possible to maximize the number of keys negotiated during the T_{MIN} interval. In fact, if a node is not able to negotiate a pairwise key with a neighbor node, it cannot communicate directly with it and this may imply an increase of energy consumption deriving from the resulting use of multi-hop communication.

Starting from the solutions adopted by LEAP+ a new handshake has been proposed to reduce the number of packets exchanged and the duration of the key negotiation phase.

3.1 Hello flag

As discussed above, the pairwise key negotiation of the LEAP+ protocol [9] is composed of various phases (Fig. 1) but starts with the $HELLO$ message broadcast. The $HELLO$ message is sent periodically during the time interval T_{MIN}.

When the nodes of the network are activated, a large amount of traffic due to the broadcast of the $HELLO$ messages and to the subsequent ACKs is generated. For instance, in a network with $n = 50$ nodes, $T_{MIN}= 30s$ and $T_{HELLO}= 3s$, the protocol generates $n \cdot (T_{MIN}/T_{HELLO}) = 500$ $HELLO$ packets. For each $HELLO$ message each node answers with an $ACK1$ message which

is received by all other nodes that are in the communication range. The communication modules of the receiving nodes must perform the basic operations to identify if they are recipients of the $ACK1$ message, regardless of whether the communication is unicast or broadcast. This may be exploited to reduce the number of $HELLO$ messages in the network by simply interpreting a generic $ACK1$ message with a destination address that is different from the receiver address as a $HELLO$ message. A $HELLO$ flag was added to the header of the $ACK1$ packet in order to enable or disable this feature. The $HELLO$ flag is necessary when a node is no longer in possession of the master key. In this case the node is not able to initiate a handshake procedure since it cannot verify the authenticity of the $ACK1$ replies. Therefore, the $HELLO$ flag must be set to $false$. Since the simulation experiments have shown that the variation of the quantity and frequency of the $HELLO$ messages have a significant impact on performance in terms of time required for the negotiation of the keys, a period $T_{HELLO}= 0.33 \cdot T_{MIN}$ was also applied to the $ACK1$ messages that have the $HELLO$ flag set to $true$.

When a node is deployed and the LEAP+ protocol starts the handshake, a $HELLO$ message is scheduled in the interval of time $0 \div T_{HELLO}$. If the node receives a $HELLO$ message from another node before sending its own $HELLO$ message it answers with an $ACK1$ message in which the $HELLO$ flag is set to $true$. The $HELLO$ message which has been replaced by the $ACK1$ with the $HELLO$ flag set to $true$ is rescheduled by a time equal to T_{HELLO}. Furthermore, a *proximity threshold* was introduced to discard and replace a $HELLO$ message, which was scheduled for an instant of time that falls within the threshold, with an $ACK1$ message. This threshold makes it possible to anticipate the beginning of the handshake through the dispatch of an $ACK1$ message that must be sent in any case. The implementation of this mechanism requires the capability to disable the incoming packets filter which discard packets that are not meant for the node. In the experimental phase a proximity threshold of $0.1 \cdot T_{HELLO}$ was adopted. Tests showed that this mechanism reduces the number of $HELLO$ message produced by the protocol.

3.2 Composite ACK1

The security of LEAP+ protocol is based on the assumption that only the possessors of the master key can authenticate the $ACK1$ through the computation of $MAC(ID_u|ID_v)_{K_v}$. The presence of ID_u in the MAC argument is critical from the security point of view because it prevents potential reply attacks. Since each node performs this computation for each $HELLO$ message received, the number of packets in the network during the handshake and the power consumption depend on the node degree distribution of the network. In networks with high average node degree the performance of the handshake may suffer from a high number of collisions and from resulting retransmissions.

Starting from these considerations a new typology of $ACK1$ packet called *composite* $ACK1$ ($ACK1_C$ for brevity) was proposed. The $ACK1_C$ packet is a special $ACK1$ packet that contains the ID_u of every node from which the

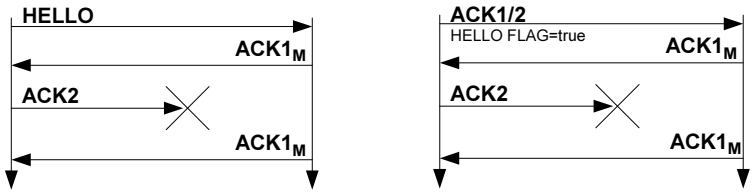
sender received a $HELLO$ message, with the corresponding MAC. The $ACK1_C$ ensures the same security features of the $ACK1$ packet but is able to manage all the pending handshake initiation requests with a single message. Only a node in possession of the master key can generate the $ACK1_C$ message and the recipients can verify the authenticity of such a packet through the same mechanisms adopted for the $ACK1$ message. Each $ACK1_C$ packet can carry a maximum number S of node IDs for which S slots are reserved $ID_{u_1} \dots ID_{u_S} | ID_v$. In this way the maximum dimension of the packet is constant.

The adoption of the $ACK1_C$ packet reduces the total number of messages required for key negotiation. Theoretically, the reduction of $ACK1$ messages is equal to $1/S$. This reduction also lowers the workload and the memory occupation required for the generation of the corresponding MAC. On the other hand, there is a limited increase in the size of the message which does not significantly affect the processing time for the computation of the MAC. The $ACK1_C$ packets are broadcast in the network as a response to multiple $HELLO$ message received by a node. Each node that receives an $ACK1_C$ packet verifies whether its own ID_u is contained in one of the slots of the packet and decides to drop it or further develop the handshake protocol. If the $HELLO$ flag is set to *true* and if the receivers have not yet exchanged a pairwise key with the sender, the $ACK1_C$ packet is interpreted as a $HELLO$ message. Otherwise the receiver verifies the authenticity of the message and continues the handshake described in the LEAP+ protocol.

When a node receive a $HELLO$ message or an $ACK1_C$ with the $HELLO$ flag set to *true* and no $ACK1_C$ is in the outgoing queue, it schedules a new $ACK1_C$ packet and randomly chooses a backoff time from the interval $0 \div T_{BACKOFF}$. If the node receives further $HELLO$s from other nodes it adds new ID_u in the free slots of the scheduled $ACK1_C$ packet until there are no more free slots or the backoff timer elapse. After sending the message, for each node whose ID_u has been added to the $ACK1_C$ packet the node awaits the $ACK2$ replies for a certain amount of time. If some of the replies are missing, the free slots of the next $ACK1_C$ scheduled in the outgoing queue are filled with the ID_u of the nodes associated to the missing replies and with the ID_u of the $HELLO$s received in the meanwhile. In order to maximize the number of useful slots of the scheduled $ACK1_C$, the IDs of the nodes with missing $ACK2$ replies near to the expiration threshold are also added in the free slots of the $ACK1_C$ message.

3.3 Modified handshake protocol

Since $ACK1_C$ packets have the same security features of $ACK2$ packets but allow multiple destinations, they can replace them. This led to a new version of the handshake protocol: $HELLO \longrightarrow ACK1_C \longrightarrow ACK1_C$. With this handshake it is possible to obtain improved performance especially in case of high-density networks and short key setup time T_{MIN}. These improvements cover the cases highlighted as critical by the security analysis presented in the previous section. In order to discriminate between $ACK1_C$ packets that require authentication from nodes in possession of the master key and $ACK1_C$ packets which replace



(a) LEAP+ (b) Proposed approach

Fig. 2. $ACK1$ retransmission in the handshake protocol.

$ACK2$ packets, an additional flag was introduced in the $ACK1_C$ packet for each slot ($ACK2$ flag). This modified handshake still guarantees that the nodes that terminate the key negotiation are authorized nodes in possession of the master key.

Since one $ACK2$ flag is associated to each slot, both $ACK1$ and $ACK2$ acknowledgment messages can coexist in the same $ACK1_C$ packet. Therefore, when a node receive an $ACK1_C$ packet it verifies if its ID_u is present in one of the slots and then checks the status of the corresponding $ACK2$ flag and of the $HELLO$ flag in order to correctly interpret the message. In the case of $ACK1_C$ packet interpreted as $ACK2$ the node generates the appropriate pairwise key and the handshake terminates. Otherwise, after the generation of the pairwise key the node dispatches an $ACK2$-like packet ($ACK1_C$ with $ACK2$ flag or $ACK2$).

Summing up, there are different handshake configurations in which each node may be involved. The negotiation procedure starts with a $HELLO$ packet or with an $ACK1_C$ packet with the $HELLO$ flag set to *true*. After the first step, the receiving nodes reply with an $ACK1_C$ packet. Then, the handshake is terminated with an $ACK2$ packet in case the sender no longer has the master key. Otherwise, the handshake is terminated with an $ACK1_C$ packet which improves the efficiency of the protocol. If a node does not receive an $ACK2$-like packet it must retransmit the $ACK1_C$ packet (Fig. 2). In Fig. 3 the possible handshake configurations are summarized. The proposed handshake requires three different packets: $HELLO$, $ACK1_C$ and $ACK2$. These packets contain the fields shown in Tab. 1.

4 Comparison between LEAP+ and the proposed approach

In this session the performance of LEAP+ and of the proposed handshake have been analyzed and compared for different network configurations. The *NS2* network simulation software has been adopted to collect large quantity of data. This software has been integrated with specific libraries for the analysis of Wireless Sensor Networks. The network parameters that was taken into account for the configuration of the simulator are:

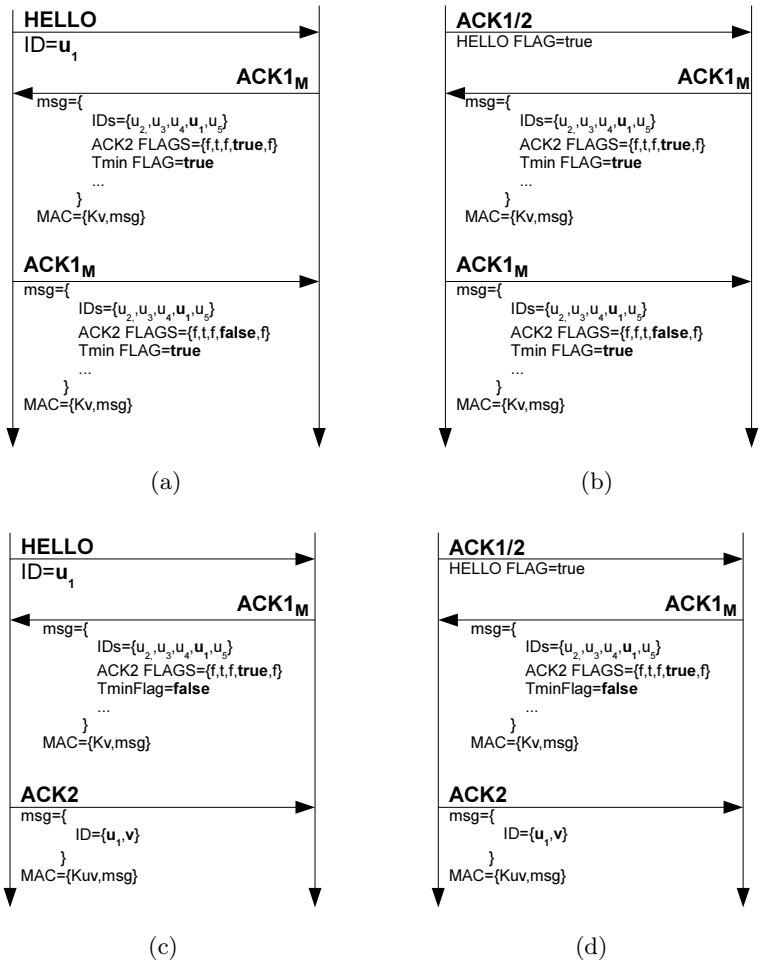


Fig. 3. Possible handshake configurations for pairwise key negotiation.

- NODES: number of active nodes in the network;
- AVERAGE NODE DEGREE: average number of nodes in the wireless communication range of each node;
- X,Y: dimension of the deployment area ($X \cdot Y m^2$ with $X = Y$);
- T_{MIN}: lifetime of the master key; after T_{MIN} elapses the node erase the master key and all the keys derived from it except its own private master key;
- DEPLOY INTERVAL: maximum time interval between the deployment and the activation of a node;
- T_{HELLO}: time interval between two consecutive $HELLO$ messages;
- $T_{BACKOFF}$: maximum time interval between the reception of a $HELLO$ message and the forwarding of the $ACK1$ reply.

Table 1. Fields contained in the packets.

Packet	Field	Description	Size (bits)
$HELLO$	NodeID	ID of the sender	16
$ACK1_C$	NodeID	ID of the sender	16
	NodeIDslots	IDs of the recipients	$16 \cdot S$
	NodeIDR	ID of the recipient	16
	Hello Flag	If true the message can be interpreted as $HELLO$	1
	T_{MIN} Flag	If true the sender has the master key and can still receive $ACK1_C$ messages as acknowledgment to $ACK1_C$ messages	1
	Ack2 Flags	If set to *true* the node with ID equal to the one contained on the corresponding slot will not send an $ACK2$ message since the $ACK1_C$ terminates the handshake	$16 \cdot S$
	Mac	Message Authentication Code obtained with the master key of the node	256
$ACK2$	NodeIDS	ID of the sender	16
	NodeIDR	ID of the recipient	16
	Hello Flag	If set to *true* the message can be interpreted as $HELLO$	1
	Mac	Message Authentication Code obtained with the private master key of the recipient node	256

4.1 Key setup time analysis

The goal of the proposed approach is to lower the time T_{MIN} so that the period of vulnerability of the nodes is reduced. In order to evaluate the performance of the modified handshake, the number of negotiated pairwise keys have been estimated for different values of T_{MIN}. A completion percentage equal to 100% corresponds to the negotiation of all the pairwise keys. The minimum values of T_{MIN} which guarantee a completion percentage of 99% have been shown in Fig. 4 for different values of the average node degree. It can be noticed that the adoption of the proposed handshake significantly reduces the time T_{MIN} by a factor that depends on the network configuration and on the protocol parameters (i.e.: the number of slots in the $ACK1_C$ packet, etc...). For the configuration shown in Tab. 2 the reduction of T_{MIN} is greater than 30% respect to LEAP+.

Data collected through the simulations showed the better scalability of the proposed approach due to the reduction of packets exchanged during the handshake. This parameter may be further reduced determining an adequate number of available slots in the $ACK1_C$ packet, thus acting on the trade-off between the packet overhead and the performance. Furthermore, a detailed analysis on the relationship between average node degree and T_{MIN} has been performed. From results presented in Fig. 5 it can be noticed that for low values of T_{MIN} and high

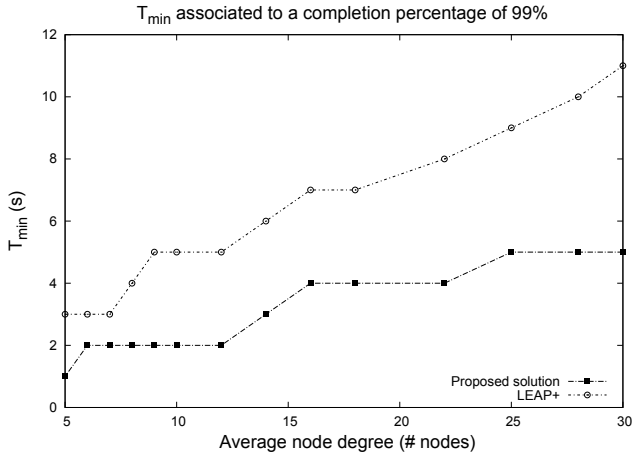


Fig. 4. T_{MIN} in the case of completion threshold equal to 99%.

Table 2. Network configuration.

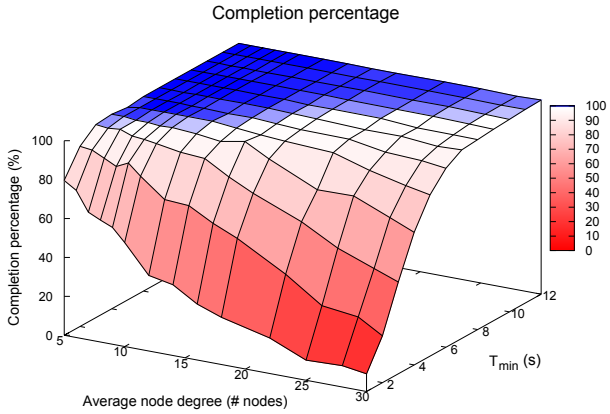
Network parameters			
NODES	30	AVERAGE NODE DEGREE	5-30
X=Y	200-900 m	T_{MIN}	1-12 s
DEPLOY INTERVAL	0 s	T_{HELLO} INTERVAL	$T_{min} \cdot 0.33$
$T_{BACKOFF}$	T_{HELLO}	NUMBER OF SLOTS S	5

values of average node degrees the percentages of completion are lower. However, the proposed approach improves these percentages in each critical configuration (see Fig. 5). For instance, in the case of $T_{MIN} = 4s$ the completion percentage of LEAP+ original handshake is about 70% whereas the completion percentage of the proposed handshake is about 100%.

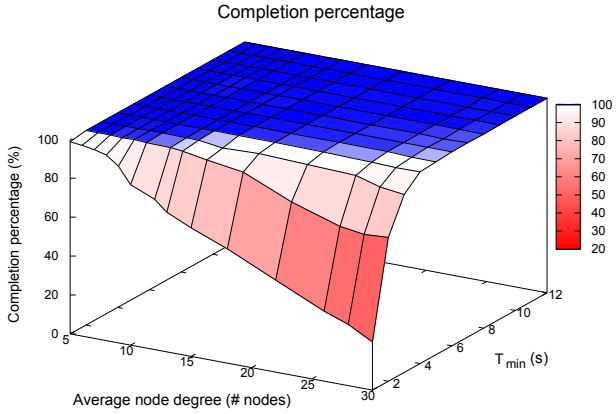
From the charts shown in Fig. 5 it can be highlighted that very short T_{MIN} intervals do not allow the negotiation of all the pairwise keys. Adopting the network parameters of Tab. 2 the LEAP+ handshake allows a completion percentage of 95% with $T_{MIN} = 6s$ while the proposed approach performs better, allowing the same completion percentage with $T_{MIN} = 3s$.

4.2 Deployment time analysis

In order to carry out a detailed analysis of the proposed handshake for different values of the deploy interval, a network configuration with high average node degree has been adopted (Tab. 3). In fact, in networks with high average node degree the activation of a large number of neighbor nodes causes the generation of a large number of packets in a limited time interval. In this context, the high number of collisions in the communication channel dramatically increases



(a) LEAP+



(b) Proposed approach

Fig. 5. Completion percentage as a function of T_{MIN} and of the average node degree.

the number of resent packets, thus stretching the negotiation time. Therefore, this condition amplifies the differences between the two handshakes and allows an effective comparison. As shown in Fig. 6, both LEAP+ and the proposed approach present weak performance for low values of the deploy interval.

However, the adoption of $ACK1_C$ packets and of the modified handshake $HELLO \longrightarrow ACK1_C \longrightarrow ACK1_C$ makes it possible to increase considerably the percentage of keys negotiated in the network, keeping the deploy interval constant. This statement is also endorsed by the study of the average number of packets received and sent by each node. In fact, the proposed handshake allows

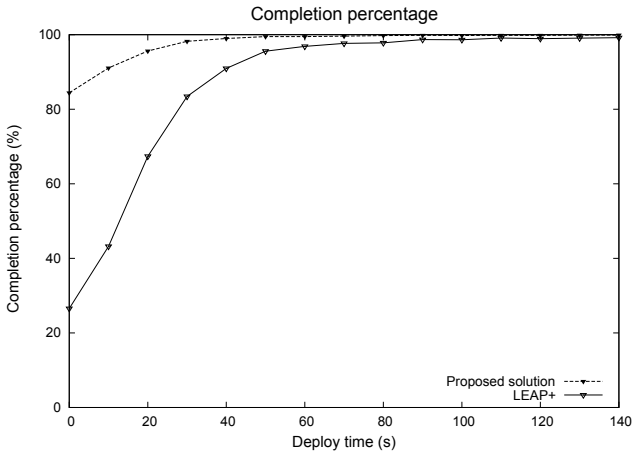


Fig. 6. Completion percentage as a function of the deploy time. Network with high average node degree .

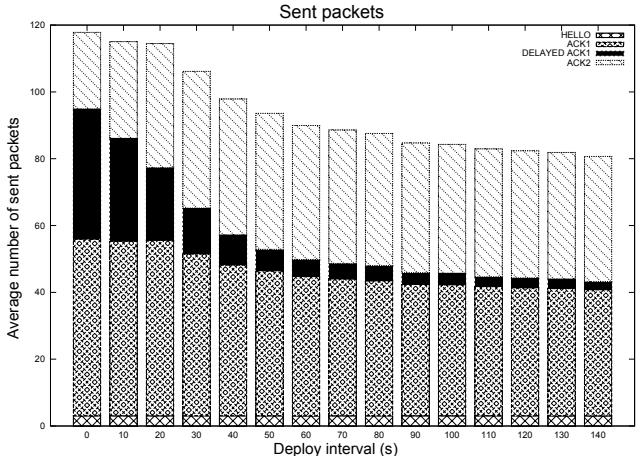
Table 3. Network configuration.

Network parameters			
NODES	70	AVERAGE NODE DEGREE	70
X=Y	140-160 m	T_{MIN}	12 s
DEPLOY INTERVAL	0-140 s	T_{HELLO} INTERVAL	$T_{min} \cdot 0.33$
$T_{BACKOFF}$	T_{HELLO}	NUMBER OF SLOTS S	5

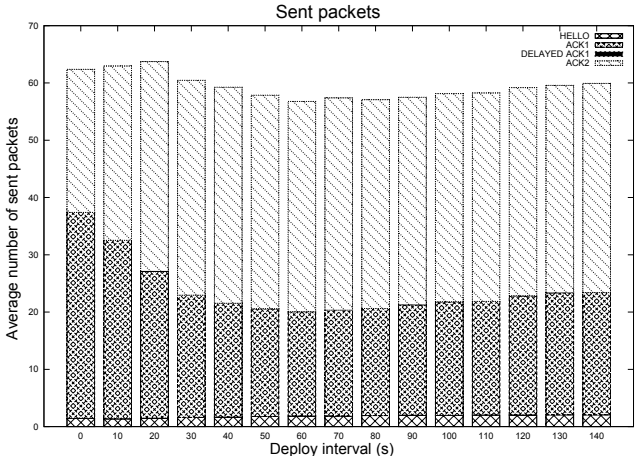
the reduction of the number of sent packet for each deploy interval, as shown in Fig. 7.

From the analysis of Fig. 7(a) it can be noticed that the high number of retransmissions is due to the loss of $ACK2$ packets. The number of retransmissions and collisions decreases as the deploy interval increases. However, the average number of received packets per node increases in the new implementation (see Fig. 8) since the $ACK1_C$ packets introduced in the proposed handshake are broadcast and potentially received by S nodes.

It is worth noting that the quantities of packets sent and received as a function of the deploy interval, shown in the previous histograms, refer to different completion percentages (data shown in Fig. 6). As highlighted in Fig. 7(a), a high number of sent packets does not necessarily implies a high number of negotiated keys. This is due to the increment in the number of collisions that occurs when the number of packets exchanged in the communication channel increases.



(a) LEAP+

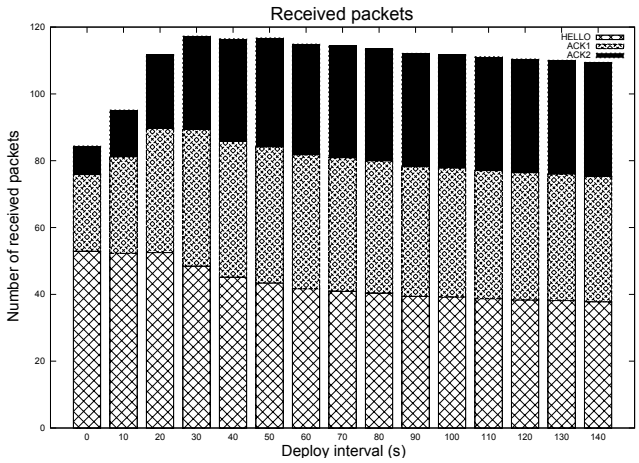


(b) Proposed approach

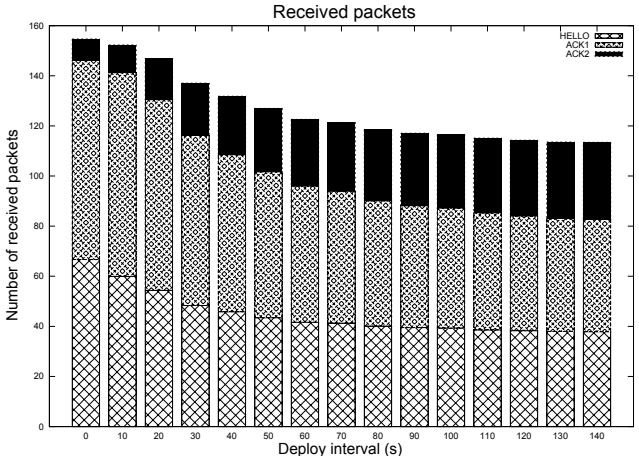
Fig. 7. Average number of packets sent by a node.

5 Conclusion

This paper presented an enhanced version of the LEAP+ protocol which improves the security of the handshake for pairwise key negotiation. The improvement consists in the reduction of the vulnerability time during which an attacker may stole the master key that is critical for the security of all the pairwise key communications. The results, obtained through a network simulator, showed significant improvements of performance in terms of reduction of the key setup time and of number of packets exchanged for the key negotiation. The improvements were more evident in the most critical contexts for the LEAP+ protocol such



(a) LEAP+



(b) Proposed approach

Fig. 8. Average number of packets received by a node.

as high density networks with low activation time. The higher efficiency in the pairwise key negotiation made it possible to shorten the interval of vulnerability of the nodes thus increasing the security of the entire network. The study carried out showed the importance of the selection of proper network configuration parameters and algorithms parameter. The evaluation of optimal values for these parameters as a function of specific constraints of the network will be the subject of future research.

Acknowledgment

This work was supported in part by grant "Nano-materials and -technologies for intelligent monitoring of safety, quality and traceability in confectionery products (NAMATECH)" from Regione Piemonte, Italy.

References

1. X. Hu, B. Wang, and H. Ji, "A wireless sensor network-based structural health monitoring system for highway bridges," *Computer-Aided Civil and Infrastructure Engineering*, vol. 28, no. 3, pp. 193–209, 2013.
2. S. Sultan, T. Khan, and S. Khatoon, "Implementation of hvac system through wireless sensor network," in *Proceedings of the 2010 Second International Conference on Communication Software and Networks*, ser. ICCSN '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 52–56.
3. I. Bekmezci, *Wireless Sensor Networks: A Military Monitoring Application.* Saarbrücken, Germany: VDM Verlag, 2009.
4. N. Gura, A. Patel, A. W, H. Eberle, and S. C. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," in *Workshop on Cryptographic Hardware and Embedded Systems 2004*, 2004, pp. 119–132.
5. K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime," in *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, ser. SASN '06. New York, NY, USA: ACM, 2006, pp. 169–176.
6. J. Zhang and V. Varadharajan, "Review: Wireless sensor network key management survey and taxonomy," *J. Netw. Comput. Appl.*, vol. 33, no. 2, pp. 63–75, Mar. 2010.
7. S. Stelle, M. Manulis, and M. Hollick, "Topology-driven secure initialization in wireless sensor networks: A tool-assisted approach," in *Proceedings of the 2012 Seventh International Conference on Availability, Reliability and Security*, ser. ARES '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 28–37.
8. L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM conference on Computer and communications security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 41–47.
9. S. Zhu, S. Setia, and S. Jajodia, "Leap+: Efficient security mechanisms for large-scale distributed sensor networks," *ACM Trans. Sen. Netw.*, vol. 2, no. 4, pp. 500–528, Nov. 2006.
10. J. Deng, C. Hartung, R. Han, and S. Mishra, "A practical study of transitory master key establishment forwireless sensor networks," in *Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks*, ser. SECURECOMM '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 289–302.
11. S. Zhu, S. Setia, and S. Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM conference on Computer and communications security*, ser. CCS '03. New York, NY, USA: ACM, 2003, pp. 62–72.
12. C. Lim, "Leap++: A robust key establishment scheme for wireless sensor networks," in *Distributed Computing Systems Workshops, 2008. ICDCS'08. 28th International Conference on.* IEEE, 2008, pp. 376–381.