Fast hierarchical key management scheme with transitory master key for wireless sensor networks

Fast Hierarchical Key Management Scheme with Transitory Master Key for Wireless Sensor Networks

Filippo Gandino, *Member, IEEE,* Renato Ferrero, *Member, IEEE,* Bartolomeo Montrucchio, *Member, IEEE,* and Maurizio Rebaudengo, *Senior Member, IEEE,*

Abstract—Symmetric encryption is the most widely adopted security solution for wireless sensor networks. The main open issue in this context is represented by the establishment of symmetric keys. Although many key management schemes have been proposed in order to guarantee a high security level, a solution without weaknesses does not yet exist. An important class of key management schemes is based on a transitory master key. In this approach, a global secret is used during the initialization phase to generate pair-wise keys, and it is deleted during the working phase. However, if an adversary compromises a node before the deletion of the master key, the security of the whole network is compromised. In this paper, a new key negotiation routine is proposed. The new routine is integrated with a well-known key computation mechanism based on a transitory master secret. The goal of the proposed approach is to reduce the time required for the initialization phase, thus reducing the probability that the master secret is compromised. This goal is achieved by splitting the initialization phase in hierarchical sub-phases with an increasing level of security. An experimental analysis demonstrates that the proposed scheme provides a significant reduction in the time required before deleting the transitory secret material, thus increasing the overall security level. Moreover, the proposed scheme allows to add new nodes after the first deployment with a suited routine able to complete the key establishment in the same time as for the initial deployment.

Index Terms—Symmetric encryption, WSN, Key management, Transitory master key

1 Introduction

Wireless sensor networks (WSNs) are a well-known enabling technology for Internet of Things (IoT). IoT solutions that eploit WSNs are present in several application fields, such as intelligent transportation systems [1], surveillance [2] and health-care [3]. Since WSNs are composed by low-cost and low-power devices, they involve several issues (e.g., MAC [4], error correction [5] and energy consumption [6]). Moreover, WSNs have relevant differences according to their application, so they require specific solutions (e.g., event-driven applications [7] and real time applications [8]).

A strict requirement for many IoT solutions is represented by data security. A strict requirement for many IoT solutions is represented by data security. Many specific solutions are needed due to the heterogeneity of IoT systems, from smart grid [9], [10] to smart home [11]. Solutions designed for WSNs should consider that this technology is exposed to many threats such as eavesdropping, hardware tampering and false messages. The majority of WSNs use *symmetric cryptography* to protect security [12]. However, this strategy requires that two nodes in the reciprocal communication range share a common *key* that will be used for the encryption and decryption of the messages and/or for their authentication. The establishment of symmetric keys is called *key management* [13], [14]. Although there are some systems to detect compromised nodes and recover security [15], [16], it is fundamental to limit the effects of possible attacks.

Many schemes have been proposed in order to establish keys in WSNs. Two elementary key management schemes are:

- Plain global key (PGK), with only one key used by all the nodes;
- Full pairwise keys (FPWK) [17], in which each node shares a specific key with each other node, so any possible link has its own key.

In these approaches all the keys are predistributed, so they do not require neither the execution of any operation nor any additional information such as deployment knowledge. PGK limits the memory overhead, but it guarantees a low level of security, since an adversary that compromised a key can eavesdrop on all the links and can introduce fake nodes in the network. FPWK provides a higher level of security, since an adversary that compromised a key can only eavesdrop on one link. However, it requires a large amount of memory, since each node must store a key for every other node in the network. Therefore, FPWK can only be used for small networks while PGK provides a low level of security.

Many key management schemes with more advanced features have been proposed in the literature. They can be divided into different families, according to their main characteristics. Some schemes are specifically designed for static networks. A well-known family of key management

- *F. Gandino, R. Ferrero, B. Montrucchio and M. Rebaudengo are with the Dipartimento di Automatica e Informatica, Politecnico di Torino E-mail: see http://www.polito.it/search/index.php?lang=en*

schemes for static networks is based on the *transitory master key* [18], [19]. In these schemes the *master key* (MK) is a common secret, which is known by all the nodes. Each pair of nodes use MK to protect the generation of their pairwise key. MK is deleted at the end of the *initialization* phase, but if MK has been compromised, all the keys in the network are compromised. The main scheme based on transitory master key is LEAP+ [18]. An important characteristic of WSNs is the possibility to add new nodes after the initial deployment (feature called *node adding*). In LEAP+, it is possible to add new nodes, since for the key establishment it is required that at least one of the involved nodes is within the initialization phase (i.e., just deployed), but the other node can also be within the following phase, i.e. the *working phase*.

This paper presents the *Hierarchical scheme with transitory master key* (HSTMK), a new key management scheme based on the transitory master key approach. The new approach is designed for static wireless sensor networks with node adding property and without deployment knowledge. The goal of HSTMK is to reduce the time window in which an adversary can achieve important secret material from the memory of the nodes of the network. The main novelty of this scheme is an efficient organization of the handshake routine, which is subdivided into separate phases. This solution reduces the number of packets exchanged during each phase. The lower number of packets produces a reduction in the number of collisions and allows a shorter key setup time. Moreover, the transitory secrets are organized in a hierarchical way, and each node can delete a part of the most dangerous transitory secrets at the end of each phase. The main benefit of HSTMK corresponds to a higher level of security than LEAP+, due to the reduction in the delay before deleting the transitory secret material. The relevance of this reduction is supported by comparing the achieved experimental delay results with data provided by an experimental analysis on node compromising [20]. The proposed analysis shows how node compromising attacks, feasible in LEAP+, are unfeasible with the proposed approach. Another benefit of HSTMK corresponds to the lower communication and computation overheads. A preliminary version of the proposed approach was presented in [21]. With respect to it the scheme has been modified and improved, in particular to allow node adding after the initial deployment. The description of the scheme has been totally modified in order to present it with more details. New theoretical and experimental analyses has been conducted and the proposed scheme has been implemented and tested on a real WSN.

The organization of the rest of the paper is as follows: in Sect. 2 related work is described. In Sect. 3 the proposed scheme is presented. In Sect. 4, the proposed approach is evaluated and compared with state-of-the-art schemes, and in Sect. 5 some conclusions are drawn.

2 Related Work

Many techniques for the establishment of the common secret in WSNs have been proposed [14], [17], [22]. This section divides the key management schemes into families and describes the most relevant ones.

2.1 Random Key Distribution

Random key distribution consists in the random distribution to each node of some secrets selected from a large pool. Two nodes can establish a link only if they share a common secret. If two neighboring nodes do not share at least a secret, they cannot communicate, so this approach reduces the connectivity. However, if a node is compromised, an adversary reaches only a limited quantity of secret material.

A well-known approach based on Random key distribution, hereinafter called EG, has been presented by Eschenauer and Gligor [23]. Before the deployment, a large pool composed by p keys is generated and a ring, composed by r keys randomly picked up from the pool, is assigned to each node. After the deployment, each node checks if its neighboring nodes share at least a common key with it. The values of p and r strongly affect the scheme performance. If r is close to p, each node has the majority of the keys and it is probably able to establish a link with all its neighboring nodes. However, if a node is compromised, the adversary achieves the majority of the keys in the pool and almost the whole network is compromised. If r is too small with respect to p, a node probably cannot establish a link per neighboring node, but an adversary with r keys can compromise only a small portion of the network.

A second protocol based on Random key distribution is *q-composite random key predistribution* [24], hereinafter called QC. In QC, two nodes have to share at least q keys in order to establish a link. They generate a new key by executing a hash function on the concatenation of the shared keys. QC is more robust than EG if a small quantity of nodes is compromised. However, it is more vulnerable if several nodes are compromised.

2.2 Global Master Key

The Global master key techniques exploit a MK shared by all the nodes and used as a common secret in order to generate pairwise keys.

The *Symmetric-key key establishment* (SKKE), adopted by ZigBee[1] uses a MK common to all the nodes. As a first step, node A sends a challenge (C_A), which is a random number. Node B sends back a message that includes its identifier ID_B, a new random challenge (C_B) and the Message authentication code (MAC), which is calculated according to ID_B, ID_A, C_B, C_A and to a constant number called k_1. The nodes execute a keyed hash function with MK as a key on the two IDs concatenated to the two random numbers as an input string. The result represents a common secret. Finally, two keys (the first one for authentication and the other one for encryption) are generated by executing a hash function on the common secret.

In SKKE, the required pre-deployment storage is small. However, also the level of security is low, since if MK is compromised an opponent can generate all the keys in the network and potentially eavesdrop on all the links.

2.3 Transitory Master Key

The Transitory master key technique consists in the use of a global secret, MK, that each node has to delete after a

1. ZigBee Specification 1.0, June 2005, ZigBee Alliance

timeout. This approach is based on the assumption that a node cannot be compromised before a specific time interval. Therefore, MK should be erased before this time.

Before the deletion of MK, a node is in the *initialization phase*, while after the deletion, it is in the *working phase*. According to the basic approach a node would not be able to establish new pairwise keys after the deletion of MK. However, some schemes implement a mechanism that allows the establishment of keys between a node within the working phase and another one within the initialization phase. Without such a mechanism it would not be possible to add new nodes into the network after the first deployment.

If an opponent compromises MK, he/she would be able to decode every message that has been eavesdropped during the initialization phase and to find all the pairwise keys that have been established by that messages. Moreover, if the specific key management scheme is compliant with nodes added to the network after the initial deployment, an adversary would be able to establish a new link with all the new nodes.

The length of the timeout affects the level of security and connectivity of the network. A long timeout increases the risk that an adversary is able to compromise MK. However, with a short timeout the nodes could not have enough time to complete the key establishment with all the neighboring nodes. Therefore, the length of the timeout must be carefully selected, especially if the node density is unknown. The time required by an adversary to compromise a node depends on many factors, such as the hardware and software characteristics of the WSN and the computational power of the opponent. An example of attack is described in [20], in which the 128 Kbytes program flash of a Mica2[2] node is dumped in 30 seconds, while all the memories of the node are dumped in less than 1 minute.

The most important scheme based on the transitory master key technique is called LEAP+ [18]. LEAP+ requires the adoption of different kinds of key depending on the type of messages. However, the pairwise key establishment represents the core of the approach and the base for the generation of the other kinds of key.

A setup task is executed before the initial deployment of the network. In this task, the administrator of the network generates and loads on each node the global master key, here called initial key K_{IN}. Each node u generates its own master key $K_u = f_{K_{IN}}(u)$, where $f_K(\cdot)$ is a pseudorandom function indexed by the key K.

At the deployment, each node triggers a timer which measures the duration of the initialization phase. Within the initialization phase, a node exchanges messages with its neighboring nodes in order to negotiate the pairwise keys. Each node u periodically broadcasts a message called *Hello*, that contains its identification code ID_u . The Hello packets are sent with a period equal to T_r. Given the duration of the initialization phase, the quantity of sent Hello messages depends on T_r, i.e., to a high value of T_r corresponds a low number of Hello messages. If each node sends many Hello messages, the probability that the neighboring nodes receive one of them may increase, but the number of collisions among messages becomes higher. A proper value for T_r must be determined taking into account the network configuration parameters like the node density and the duration of the timeout.

2. Mica2, https://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf

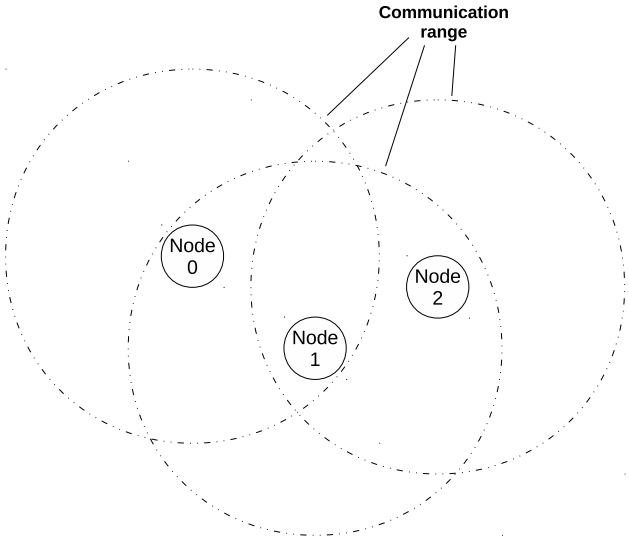

Fig. 1. Example of deployment.

When a node v receives a Hello packet, it replies with an acknowledgment message Ack. This message, which contains ID_v, is unicast and is sent back to the sender of the received Hello. In order to authenticate the packet, its sender attaches a MAC computed by using the master key K_v. In order to reduce the probability of collisions, the Ack packets are sent after a random time. After sending the Ack message, node v generates the pairwise key by computing $K_{uv} = f_{K_v}(u)$. When node u receives the Ack message, it verifies the integrity and authenticity of the message by computing the MAC. If it is correct, node u computes the master key of node v, $K_v = f_{K_{IN}}(v)$, and then the pairwise key, $K_{uv} = f_{K_v}(u)$. After this phase, the handshake is completed and both nodes share the same pairwise key K_{uv}.

When the timer of the initialization phase elapses, the node deletes K_{IN}. Without K_{IN}, the nodes are not able to start a new handshake procedure for the negotiation of a pairwise key, since they cannot compute the master keys of the other nodes. However, the nodes that deleted the initial key can still answer to the Hello messages, since only the initiator of the handshake, which sends the Hello, has to use the initial key. Therefore, in LEAP+ it is possible to add new nodes to the network after the initial deployment.

3 Hierarchic scheme with transitory master key (HSTMK)

The *Hierarchic scheme with transitory master key* (HSTMK) is a key management scheme based on a transitory global secret. The pairwise keys are computed using the same formulas used in LEAP+. All the nodes know a global secret (i.e., the initial key k_{IN}) that an adversary could try to steal in order to compromise the security of the whole network. Moreover, in order to allow node adding, each node i knows

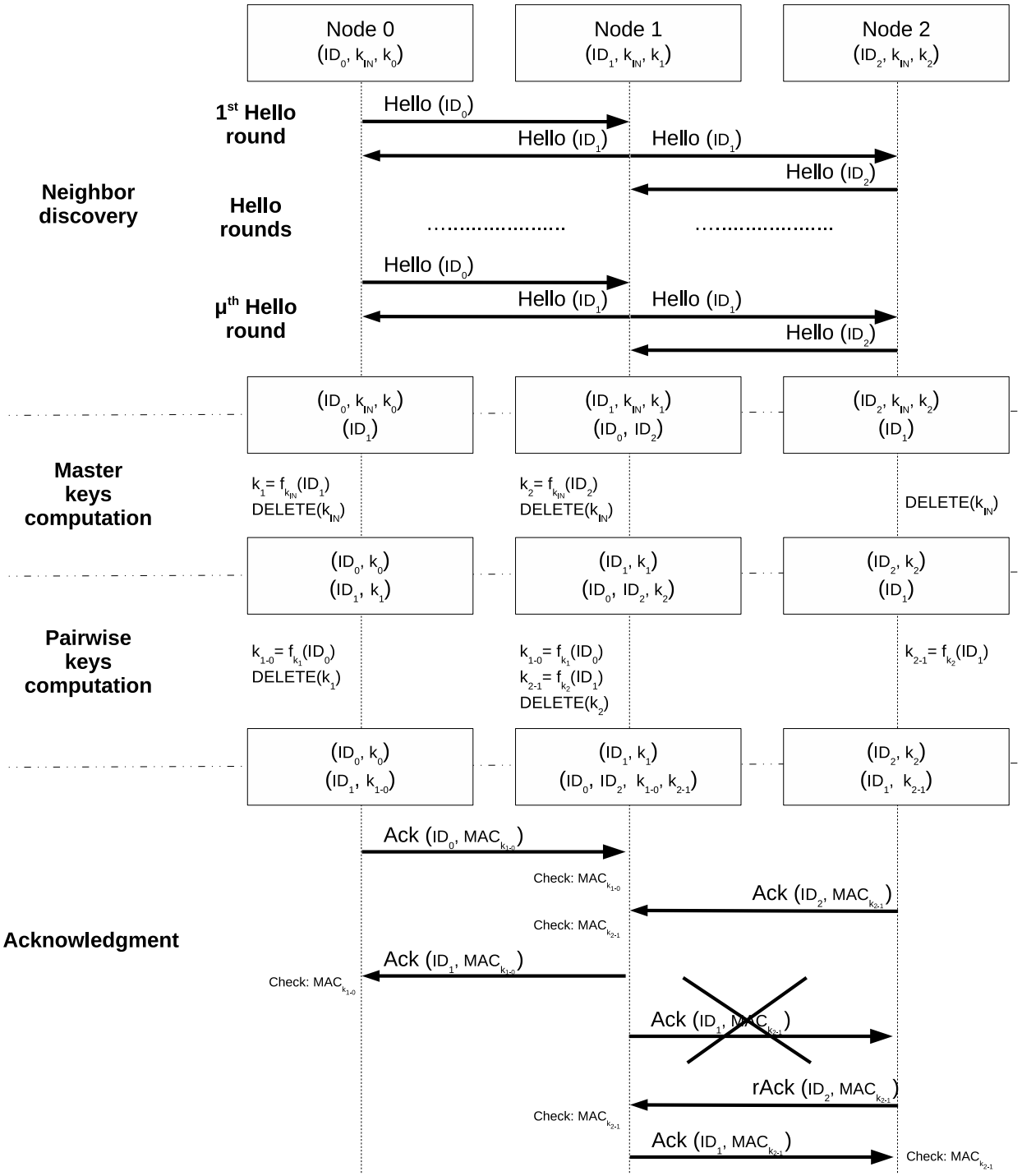

Fig. 2. Initialization phase.

a specific secret (i.e., the master key k_i) that may be used by an adversary to clone that specific node.

The two objectives of HSTMK are:

- to reduce the time spent before completing the deletion of the whole transitory secret material;
- to anticipate the elimination of the most critical part of the secret material.

In order to reach these goals, the *initialization* phase is divided into hierarchical subphases, so that each subphase has a level of security higher than or equal to the previous one. Fig. 1 shows an example of deployment with three nodes. Node 1 is within the communication range of Node 0 and Node 2, but the distance between Node 0 and Node 2 is too long to establish a link.

The subphases that compose the initialization are:

- *Neighbor discovery*: the nodes exchange Hello packets in order to identify their neighboring nodes;
- *Master keys computation*: the nodes compute the secrets of their neighboring nodes in order to compute the common pairwise keys;

- *Pairwise keys computation*: the nodes compute the final pairwise keys;
- *Acknowledgment*: the nodes send *Ack* messages in order to authenticate the key establishment.

A part of the transitory secrets is deleted at the end of the second subphase, while the rest of them is deleted at the end of the third subphase. Therefore, the fourth subphase has a security level higher than the third subphase, and the third one has a security level higher than the second one.

Before the deployment of the nodes that compose the WSN, each node is initialized with the following data:

- the identification number of the node, ID_i;
- the shared transitory global secret, k_{IN};
- and the master key of the node, k_i.

Fig. 2 shows an example of initialization with the network proposed in Fig. 1. The information held by each node are written between parenthesis in the rectangle corresponding to that node.

3.1 First initialization subphase: neighbor discovery

The first subphase is used by the nodes in order to identify their neighboring nodes. The nodes broadcast *Hello* messages which simply contain their own identification number. Each node stores the identification numbers from the received Hello messages. The first subphase is composed by μ Hello rounds. During each round, any node sends a Hello at a random moment selected between the beginning of the round and its end. The duration of a round is T_r, so, the total time of the subphase is $T_1 = \mu \cdot T_r$. The probability of interferences among simultaneous messages increases if T_r is short and if the nodes are densely deployed. A high value of μ increases the probability that a node receives at least one Hello message per neighboring node. However, high values for T_r and μ prolong the first subphase. Therefore, the parameters T_r and μ must be carefully set according to the characteristics of the network.

In Fig. 2, Node 1 broadcasts a message that contains ID_1 during every round. This message is received by the other nodes. Node 0 and Node 2 broadcast their respective messages, which are received by Node 1.

3.2 Second initialization subphase: master keys computation

In order to establish a common pairwise key, two nodes need a shared secret. Like in LEAP+, this secret is represented by a master key. Each node has a specific master key, which is written in its memory during the predeployment phase. The master key k_i of node i is calculated as:

$$k_i = f_{k_{IN}}(ID_i) \tag{1}$$

where $f_{k_{IN}}$ is a keyed pseudorandom function initialized by k_{IN} and executed on the identification number of node i. Therefore, each node is able to calculate the master key of any other node.

The pairwise keys are computed by using one of the two master keys owned by the involved couple of nodes. Therefore, considering a couple of neighboring nodes, only one node has to compute the master key of its neighboring node. The maximum quantity of keys that a node has to compute corresponds to the quantity of neighboring nodes, and the minimum corresponds to 0. In order to reduce the probability that the computational overhead due to the master key calculation is not equally distributed among nodes, a master key selection routine based on the identification numbers of nodes is adopted. The details of that routine are: (i) the identification numbers of nodes are uniformly distributed inside the range of possible numbers; (ii) the set of the n possible numbers is considered as a ring in clockwise order: the lowest number follows the highest one; (iii) considering modular arithmetic, the two subtractions between the identification numbers of a couple of neighboring nodes are considered (i.e., exchanging minuend and subtrahend), then the subtrahend that produces the lowest result is selected (iv) if both the subtractions produce the same result, the highest identification number is selected; (v) the master key of the selected node is used for the generation of the pairwise key. The selection method can be summarized by the following pseudo-code:

```
IF |(ID_i−ID_j)|_n>|(ID_j−ID_i)|_n OR
  ( |(ID_i−ID_j)|_n=|(ID_j−ID_i)|_n AND (ID_i>ID_j) )
THEN select ID_i,
ELSE select ID_j;
```

with $|\cdot|_n$ being the modulo n operation. According to the described system, the quantity of master keys that a node has to compute corresponds to a random value that follows the binomial distribution with success probability equal to 0.5 and a number of extractions equal to the amount of neighboring nodes. Therefore, this technique reduces the probability of an unfair distribution of the workload.

During the second subphase, each node checks all the stored identification numbers, and if required computes and stores the corresponding master key. After computing the master keys, the node deletes k_{IN}. If the node has not computed all the required keys before a time T_2, it stops the computation and deletes k_{IN}. This timeout is required to avoid that an adversary sends infinite fake Hello messages to a node with the goal of postponing the deletion of k_{IN}.

In Fig.2, the set of possible identification numbers is from 0 to 2. Since $|1-0|_3 = 1$ is lower than $|0-1|_3 = 2$, node 0 computes k_1 by using (1) on ID_1. In the same manner Node 1 has to compute k_2. At the end, all the nodes delete k_{IN}.

3.3 Third initialization subphase: pairwise keys computation

Each couple of nodes compute the pairwise key by executing the pseudorandom function $f_k(\cdot)$ initialized by the master key of one node and executed on the identification number of the other one. If node i and node j have selected k_i as the common secret, the pairwise key is:

$$k_{i-j} = f_{k_i}(ID_j). \tag{2}$$

According to the selected master key, a couple of nodes i and j could obtain two different pairwise keys (k_{i-j} or k_{j-i}). However, the exploited selection routine, described in Section 3.2, addresses this issue.

During the third subphase, each node computes and stores a pairwise key per neighboring node. According to the master key selection routine, the node will use its master

key or the master key of the other node, which has been computed during the second subphase. After computing all the pairwise keys, the node deletes the master keys of all the other nodes. If the node has not computed all the required pairwise keys before a time T_3, it stops the computation and deletes the master keys. Also in this case, the timeout is a protection against an adversary that sends infinite fake Hello messages to a node with the goal of postponing the deletion of the master keys. However, each node still stores its master key.

In the example shown in Fig.2, node 0 computes k_{1-0} by using k_1, which was computed during the previous subphase. Node 1 computes k_{2-1} by using k_2 and k_{1-0} by using its master key k_1. Node 2 computes k_{2-1} by using its master key k_2. After computing the pairwise keys, node 0 deletes k_1 and Node 1 deletes k_2.

3.4 Fourth initialization subphase: acknowledgment phase

Although the nodes have computed a pairwise key per neighboring node and deleted all the transitory secrets, the received Hello messages have been neither authenticated nor confirmed by Ack messages.

During the fourth and last subphase of the initialization, a node sends an Ack message per any computed pairwise key (i.e., per neighboring node). The content of an Ack message is the identification number of the sender and a *Message authentication code* (MAC) computed with the pairwise key. After receiving an Ack, if the receiver stores the corresponding pairwise key, it checks the MAC. If the message is authentic the pairwise key is confirmed. The nodes ignore Ack messages from nodes for which they have not calculated the pairwise key.

After sending all the Ack messages, if some pairwise keys have not been confirmed, the node sends a $rAck$ message (*request of acknowledgment*) per each unconfirmed key. The $rAck$ message has the same content of the Ack message. After receiving a $rAck$ message, a node checks its authenticity and answers with a Ack message. According to the density of the network, if no answer is obtained to a $rAck$ message, multiple iterations of the acknowledgment request are possible. From a security point of view, a specific timeout is not requested, since the whole transitory material has been already deleted. However, a maximum number MAX_{ack} of $rAck$ transmissions may be set, in order to save resources.

In the example, each node sends an Ack message to each neighboring node: Node 0 and Node 2 send the Ack message to Node 1, while Node 1 sends an Ack message to Node 0 and another one to Node 2. In the example, the Ack message sent by Node 1 to Node 2 is lost. Since Node 2 does not receive the Ack message, it sends a $rAck$ message to Node 1, which answers with its Ack message.

3.5 Node adding

When a new node is deployed among previously deployed nodes (i.e., in the working phase) a different scheme is executed.

The new node, starting its in the initialization phase, sends a Hello message per hello round. When a node in the working phase receives a Hello message, it answers with a working hello *WHello* (which contains its identification number) and records the received identification number.

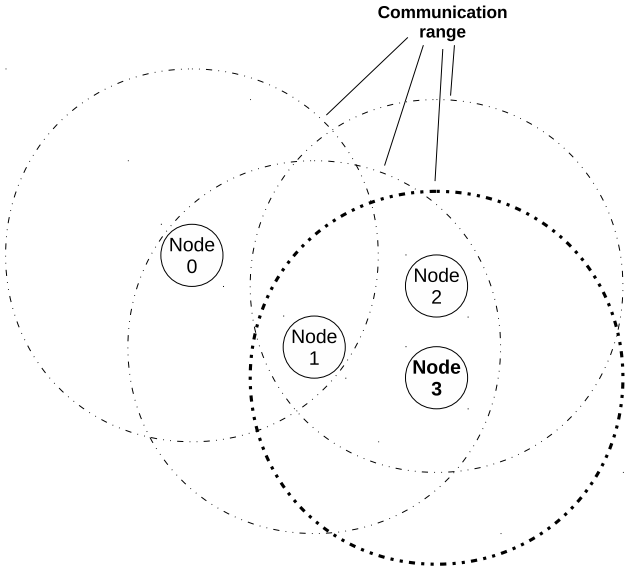

Fig. 3. Adding Node 3

In the second subphase, the master key of the node in the working phase is always selected. The other steps of the initialization are not modified.

An example of node adding is shown in Fig. 3 and Fig. 4. Fig. 3 shows the deployment with a new node, i.e., Node 3. The new node is within the communication range of Node 1 and Node 2. Fig. 4 shows the initialization of Node 3. In the first subphase, Node 3 broadcasts Hello messages that are received by its neighboring nodes, i.e., Node 1 and Node 2. Node 1 and Node 2 answer to Node 3 with a WHello message that provides the identification number of the sender and the information of being within the working phase. During the second subphase, since Node 3 knows that Node 1 and Node 2 are within the working phase, it computes their master keys, i.e., k_1 and k_2. Then, Node 3 deletes k_{IN}. During the third subphase, Node 1 and Node 2 compute the pairwise key with Node 3 by using their master key on the identification number of Node 3. Node 3 executes the same computation, in order to generate k_{1-3} and k_{2-3}, then it deletes k_1 and k_2. During the acknowledgment subphase, each node sends an Ack message per established pairwise key.

4 Evaluation and Comparison

In this section, the proposed scheme is analyzed and compared to LEAP+. An analytical study considers network and security properties, while an experimental study presents an implementation of the proposed approach and compares its time performance with an implementation of LEAP+.

4.1 Connectivity

Some key management schemes do not allow the establishment of all the links among neighboring nodes. The connectivity level represents the ratio between the number of established links and the total quantity of possible links.

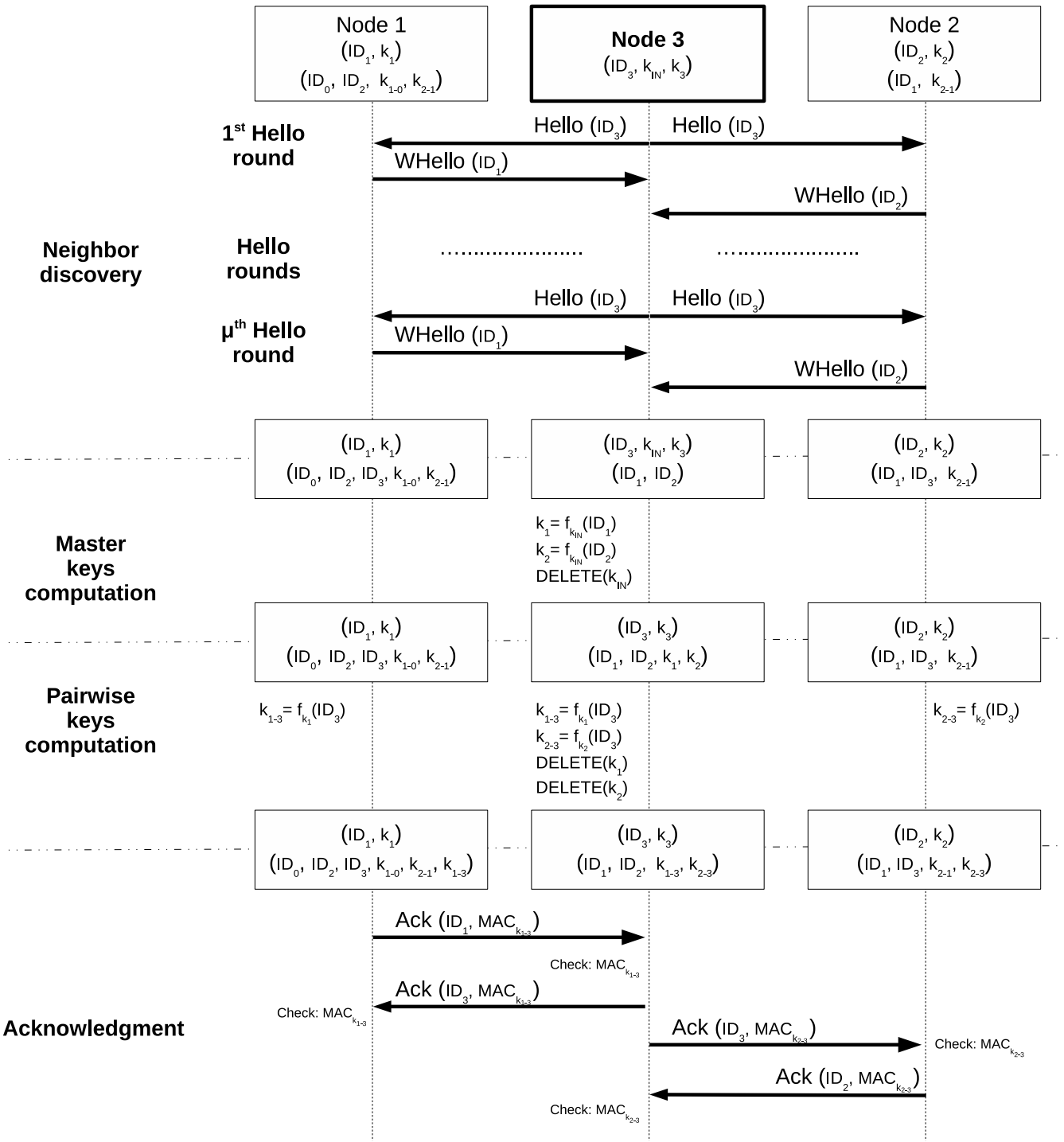

Fig. 4. Node adding.

In an ideal condition, both LEAP+ and HSTMK provide the same connectivity level. A node should be able to establish a link with all the neighboring nodes. However, in both the schemes, a too short timeout, due to the security constraints, would reduce the quantity of established links. Therefore, the time performance of the two schemes would affect the trade off between connectivity and security.

4.2 Resilience

An adversary that has compromised some nodes could try to use the achieved secret information to eavesdrop on links of the network or to pass authenticity checks. The resilience corresponds to the ability of resisting to an adversary that has compromised one or more nodes and that knows all their secret material.

If some nodes are compromised after the end of the initialization phase, HSTMK and LEAP+ provide the same level of resilience. In both the schemes, nodes store only

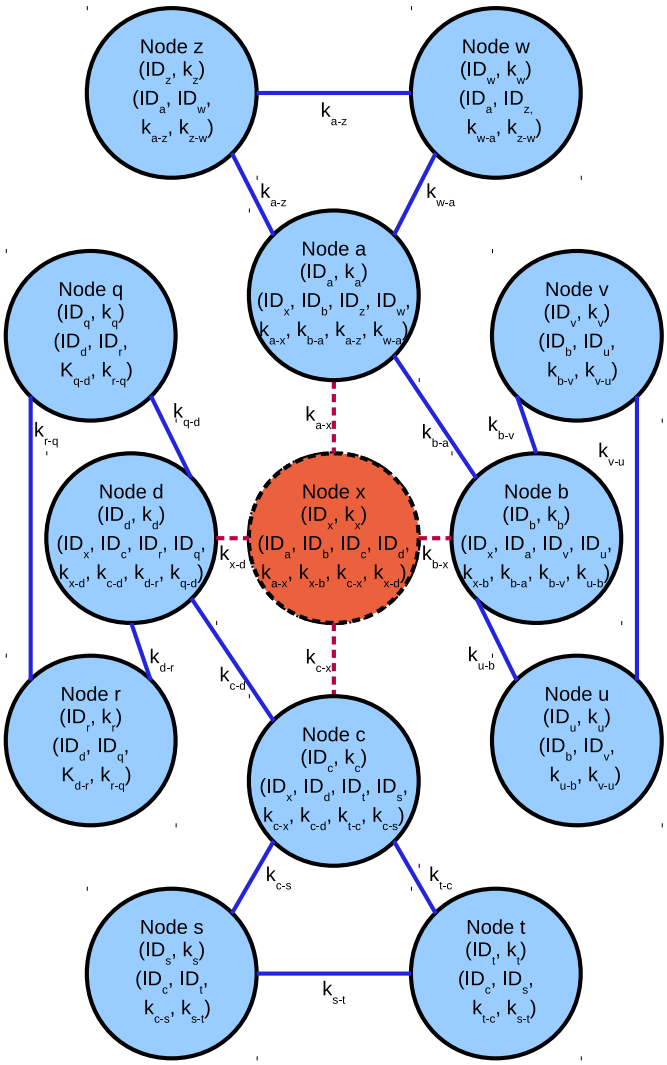

Fig. 5. Effects of a compromised node during the working phase with HSTMK and LEAP+.

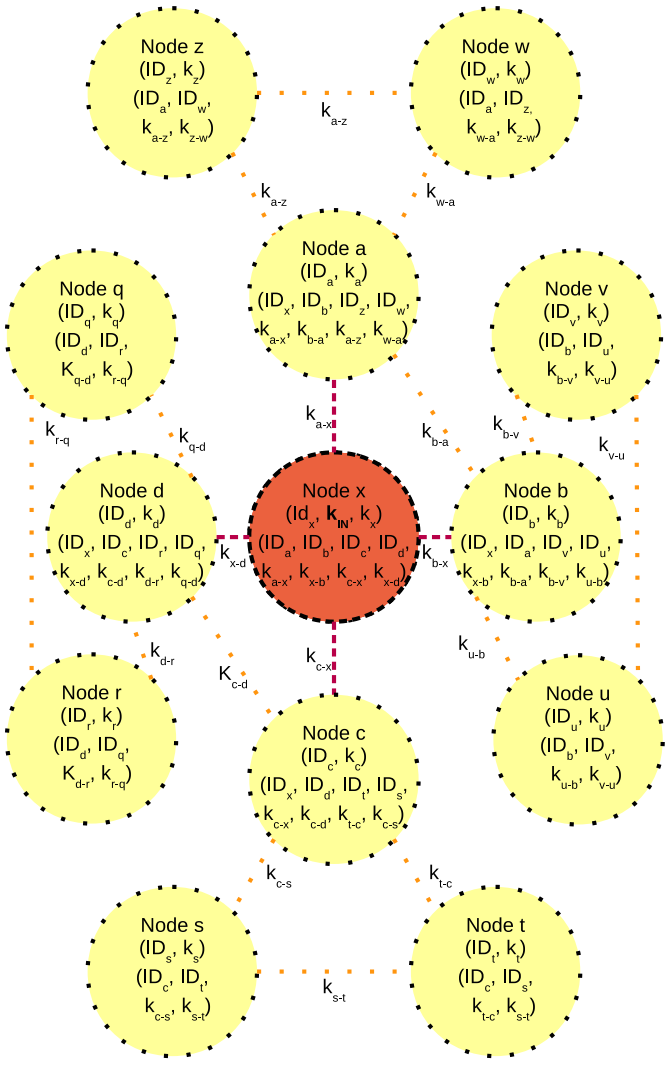

Fig. 6. Effects of a compromised node before deleting k_{IN} with HSTMK and LEAP+.

their master key and the shared pairwise keys. Therefore, an adversary that has compromised some nodes does not achieve any advantage, apart from the opportunity to impersonate the compromised nodes while interacting with their original neighboring nodes. Fig.5 shows an example of an adversary that compromises one node during the working phase. Circles with dashed borders represent compromised nodes, while circles with continuous borders represent safe nodes. Dashed lines represent links with a compromised node, while continuous lines represent safe links. The adversary has the control of Node x. The links of Node x are controlled by the adversary, but he/she can neither eavesdrop on other links, nor introduce new nodes able to pass any authenticity check.

If some nodes are compromised before the deletion of all the transitory secret material, in both the schemes the adversary can eavesdrop on all the links and be authenticated as a new original node by any authentic node. Fig.6 shows an example of an adversary that compromises one node before deleting k_{IN}. Circles with dashed borders represent compromised nodes, while circles with dotted borders represent nodes that are not able to recognize fake nodes introduced by the adversary. Dashed lines represent links with a compromised node, while dotted lines represent links that the adversary can eavesdrop on. The adversary has the control of Node x, which still stores k_{IN}. The four links of Node x are controlled by the adversary, and he/she can eavesdrop on all the other links, and introduce new nodes able to pass all the authenticity checks.

In HSTMK, there is also an intermediate state in which the adversary compromises nodes after the deletion of k_{IN}, but before the deletion of the master keys of neighboring nodes. Compromised nodes have computed the master keys of approximately half of their neighboring nodes, as described in Section 3.2, so these master keys are compromised. Since the adversary knows their master keys, he/she can introduce new nodes that will be authenticated only by those nodes. Moreover, approximately half of the links of each node are based on its master key. Therefore, the adversary will be approximately able to eavesdrop only on half of the links of the nodes of which he/she knows the master key. Fig. 7 shows an example of an adversary that compromises one node during the third initialization subphase. Circles with dashed borders represent compro-

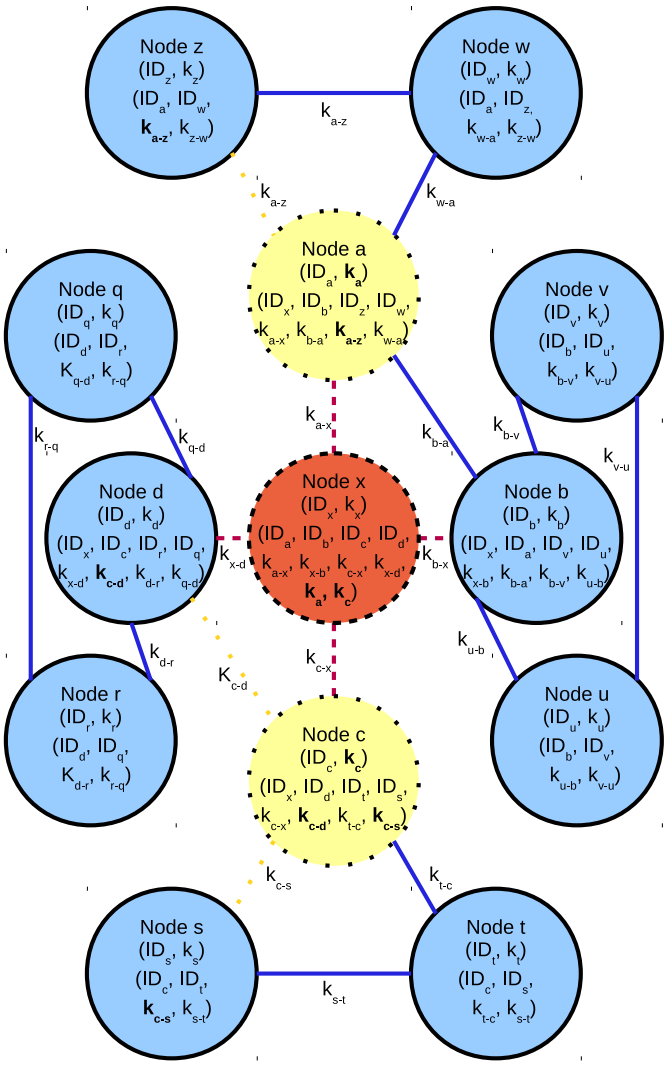

Fig. 7. Effects of a compromised node during the third initialization subphase with HSTMK.

mised nodes controlled by the adversary, circles with dotted borders represent nodes that are not able to recognize fake nodes, and circles with continuous borders represent safe nodes. Dashed lines represent links with a compromised node, dotted lines represent links that the adversary can eavesdrop on, and continuous lines represent safe links. The adversary has the control of Node x, which still store k_a and k_c. He/she controls the links of Node x. Moreover, the adversary can introduce new nodes able to pass the authenticity checks of Node a and Node c, since Node x stored their MKs when it was compromised. The adversary can also eavesdrop on the link between Node a and Node z, on the link between Node c and Node s and on the link between Node c and Node d, since the corresponding pairwise keys have been computed by using a compromised MK.

4.3 Security analysis

Since HSTMK uses the same key generation mechanism as LEAP+, the level of security is the same. The only possible issue is represented by the delayed authentication check on the received Hello messages.

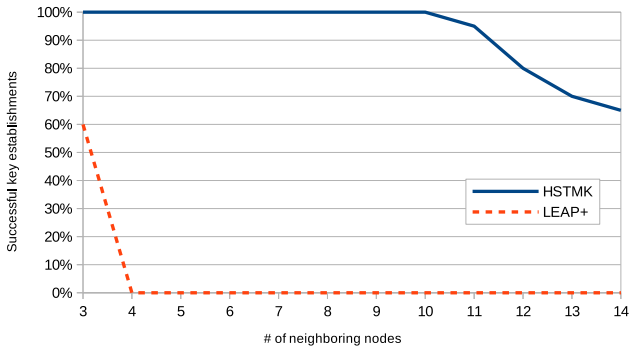

Fig. 8. Number of successful key establishment over 20 trials with $T_r = 40$ ms and $\mu = 4$, according to node density.

TABLE 1
Operations executed before deleting the secret material

Deleted Secret	Effort	LEAP+	HSTMK
k_{IN}	Sent messages	$\mu + v$	μ
	Sent data (bits)	$(\mu + v) \cdot l_{ID} + v \cdot l_{key}$	$\mu \cdot l_{ID}$
	Computation	$[1, 2]\, vf() + v$MAC	$[0, 1]\, vf()$
Master keys	Sent messages	$\mu + v$	μ
	Sent data (bits)	$(\mu + v) \cdot l_{ID} + v \cdot l_{key}$	$\mu \cdot l_{ID}$
	Computation	$[1, 2]\, vf() + v$MAC	$[1, 2]\, vf()$

In LEAP+, according to the three-way handshake implementation [18], a fake Hello is detected since either a wrong Ack2 message or no Ack2 is received. Therefore, this attack cannot compromise the security. However, by sending a fake Hello the adversary increases the amount of exchanged messages during the initialization and produces some computational overhead for the nodes. The receiving nodes will compute a pairwise key, maybe a master key, and each node will send an Ack message. Moreover, the receiving nodes have to store the pairwise key until the false authenticity is detected. Since all these operations are executed before the timeout, they can reduce the probability that nodes complete the key establishment.

In HSTMK, the fake Hello produces the same computational and communication overheads. However, differently by LEAP+, all the additional Ack messages are sent after the deletion of the transitory secrets. Therefore, in HSTMK the overheads that can affect the probability that the nodes complete the key establishment are lower than in LEAP+.

4.4 Communication and Computational Effort

The time required before deleting the secret material depends on the activities executed between deployment and deletion. Table 1 shows the communication and computational effort required by LEAP+ and by the proposed scheme. In LEAP+, when an acknowledge packet collides, a retransmission is required. In order to provide a fair comparison, the best theoretical case for LEAP+ is considered

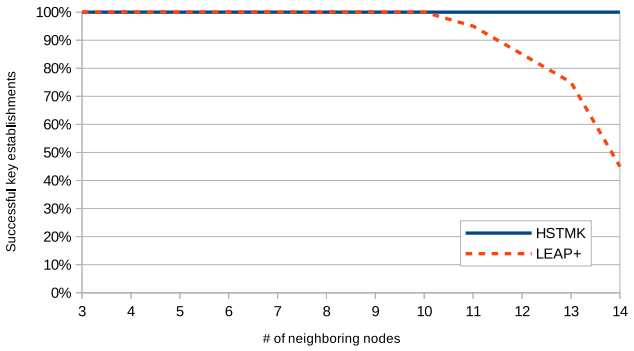

Fig. 9. Number of successful key establishment over 20 trials with $T_r = 160$ ms and $\mu = 64$, according to node density.

and no retransmission is calculated. v represents the number of nodes in direct communication, l_{ID} the length of the identifier of nodes, l_{key} the length of the adopted secret keys, $f()$ the computation of a pseudo-random function, and MAC() the computation of a message authentication code.

Let us consider the following case study: $\mu = 4$, $v = 10$, $l_{ID} = 2$ bytes and $l_{key} = 16$ bytes. Before deleting k_{IN}, a node in LEAP+ sends 14 messages composed by 188 bytes, computes 10 MAC and between 10 and 20 pseudorandom functions, while in HSTMK a node sends 4 messages composed by 8 bytes and computes between 0 and 10 pseudorandom functions.

4.5 Experimental analysis

The WSN used for the experimentation is composed by Mote Sky IV[3]. The nodes work with TinyOS[4], a open source event-driven operating system designed for low-power wireless devices. A prototype of HSTMK and an implementation of LEAP+ were developed by using NesC[5], the programming language developed for TinyOS. Different WSNs were deployed and tested in order to evaluate the proper time before deleting the secret material. While in LEAP+ all the material is deleted simultaneously, in HSTMK the deletion of k_{IN} and the deletion of the master keys of the neighboring nodes are considered separately.

The parameters of HSTMK have been described in Section 3. In LEAP+, after receiving a Hello message, nodes wait for a random time before answering, in order to decrease the probability that multiple Ack messages are simultaneously sent. Therefore, also a maximum time (T_b) before replying to a message must be set. The values used for the experiments are:

- $\mu = 4^i, i \in \mathbb{N}, i \geq 1$,
- $T_r = 4^j \cdot 10ms, j \in \mathbb{N}, j \geq 1$,
- $T_b = \frac{T_r}{4}$.

The value of T_b strongly affects the probability that Ack messages are received. The set up of this value is based on many experiments with different parameters. Since subphases 2 and 3 do not involve any external input, T_2 and T_3 need an upper bound, but the subphases are concluded immediately after completing the computation. The values of μ and T_r were set up on exponential scale (i.e., 1, 4, 16, 64, etc.). All the experiments were repeated 20 times.

Fig. 8 and Fig. 9 show the results of the test executed with $T_r = 40$ ms and $\mu = 4$, and with $T_r = 160$ ms and $\mu = 64$, respectively. These parameters have been selected in order to highlight the effects of an increasing number of nodes. The charts show the percentage of experiments in which phase 1 has been completed successfully (i.e., all the nodes established a pairwise key per link before the deletion of the secret material). It is possible to observe that by increasing the quantity of nodes in the WSN, the probability of completing the handshake in a fixed time decreases. Moreover, it is possible to observe that HSTMK is faster than LEAP+, since it is able to complete the handshake within a shorter time $T_r \cdot \mu$. Moreover, the charts highlight the importance of a proper configuration, set according to the density of the network. In fact, in Fig. 8, the values of T_r and μ are too small for LEAP+, so most of the pairwise keys are not established. The same values are a suitable choice for HSTMK if the number of neighboring nodes is close to 10, while they do not allow the completion of key establishment for a higher density. On the contrary, for a lower density these values are too high, since all the pairwise keys are established, but the deletion of the secret material is unnecessarily postponed. Similar conclusions can be applied to the chart in Fig. 9.

Fig. 10 compares the time required by HSTMK and LEAP+ before deleting the secret material used during the initialization. The experiments have been iterated with different timeout, and the chart shows the smallest time such that all the pairwise keys have been established in at least 19 experiments over 20. According to the selected parameters, the time required by LEAP+ is always equal to 10 ms multiplied by a power of 4. Therefore, it is steady for some numbers of nodes and then it increases of 4 times. The time required by HSTMK is not exactly equal to a power of 4, since the time for the computation of the keys must be added.

It is possible to observe that HSTMK is always faster. Moreover, by increasing the density, the ratio between the time before deleting the secret material in LEAP+ and in HSTMK increases from 3 to 230 times. This result highlights that HSTMK guarantees good performance even for densely deployed networks. HSTMK provides such a high level of scalability, since the separation in subphases and the reorganization of the key establishment avoid the congestion of sent messages and cryptographic computation.

Considering that the experiments presented in [20] show that the transitory secrets can be stolen in tens of seconds, the time reduction provided by HSTMK consistently reduces the risk for the transitory secrets, especially in network with a large density. Moreover, it is observed that, in HSTMK, new nodes join the network by performing the same operations at the first deployment as at subsequent ones. Therefore, new nodes can be added to the network by satisfying the same time constraints and providing the same security level.

3. Mote Sky IV, manufactured by Moteiv Corporation, http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf
4. TinyOS, http://www.tinyos.net/
5. nesC, http://nescc.sourceforge.net/

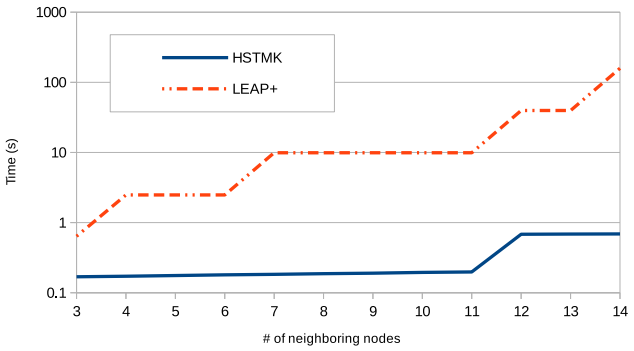

Fig. 10. Time before deleting the secret material according to the density.

5 Conclusions

This paper presented a new key management scheme based on transitory master key. The main novelty of the proposed approach is represented by a hierarchic key negotiation technique. The proposed technique improves the security and the performance. The key setup time, which is the time during which an attacker may capture the initial key, is strongly lowered in the new scheme. The results collected after extensive experiment sessions with real devices showed significant improvements of performance in terms of reduction of the key setup time and of number of packets exchanged for key negotiation. The proposed scheme also improves the security of the network in terms of vulnerability to clone attacks since the very first instants after the deployment. The improvements are remarkable especially for high density networks. The higher efficiency in the pairwise key generation process allows the improvement of the security through the decrease of the key setup time. The experimental results highlight the importance of a proper configuration for the network parameters, which must take into account the node degree of the network. In fact, a too short initialization phase would reduce the percentage of established pairwise keys, while a too long initialization phase would postpone the deletion of the secret material.

Acknowledgment

This work was partially supported by the grant Bando Smart Cities and Communities, Progetto OPLON (OPportunities for active and healthy LONgevity) funded by Ministero Istruzione, Universita' e Ricerca, Italy.

References

[1] X. Hu, L. Yang, and W. Xiong, "A novel wireless sensor network frame for urban transportation," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 586–595, Dec 2015.

[2] E. Onur, C. Ersoy, H. Deli, and L. Akarun, "Surveillance with wireless sensor networks in obstruction: Breach paths as watershed contours," *Computer Networks*, vol. 54, no. 3, pp. 428 – 441, 2010.

[3] L. Catarinucci, D. de Donno, L. Mainetti, L. Palano, L. Patrono, M. L. Stefanizzi, and L. Tarricone, "An iot-aware architecture for smart healthcare systems," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 515–526, Dec 2015.

[4] H. Yan, Y. Zhang, Z. Pang, and L. D. Xu, "Superframe planning and access latency of slotted MAC for industrial WSN in IoT environment," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 2, pp. 1242–1251, May 2014.

[5] K. Yu, F. Barac, M. Gidlund, and J. Akerberg, "Adaptive forward error correction for best effort wireless sensor networks," in *Communications (ICC), 2012 IEEE International Conference on*, June 2012, pp. 7104–7109.

[6] U. Kulau, F. Bsching, and L. Wolf, "Undervolting in wsns: Theory and practice," *IEEE Internet of Things Journal*, vol. 2, no. 3, pp. 190–198, June 2015.

[7] J. Vales-Alonso, E. Egea-Lopez, M. V. Bueno-Delgado, J. L. Sieiro-Lomba, and J. Garcia-Haro, "Optimal p-persistent MAC algorithm for event-driven wireless sensor networks," in *Next Generation Internet Networks, 2008. NGI 2008*, April 2008, pp. 203–208.

[8] E. Egea-López, J. Vales-Alonso, A. Martínez-Sala, J. García-Haro, P. Pavón-Mariño, and M. Bueno Delgado, "A wireless sensor networks MAC protocol for real-time applications," *Personal and Ubiquitous Computing*, vol. 12, no. 2, pp. 111–122, 2008.

[9] M. M. Fouda, Z. M. Fadlullah, N. Kato, R. Lu, and X. S. Shen, "A lightweight message authentication scheme for smart grid communications," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 675–685, Dec 2011.

[10] Z. M. Fadlullah, N. Kato, R. Lu, X. Shen, and Y. Nozaki, "Toward secure targeted broadcast in smart grid," *IEEE Communications Magazine*, vol. 50, no. 5, pp. 150–156, May 2012.

[11] Y. Li, "Design of a key establishment protocol for smart home energy management system," in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2013 Fifth International Conference on*, June 2013, pp. 88–93.

[12] Y. Zhou, Y. Fang, and Y. Zhang, "Securing wireless sensor networks: a survey," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 3, pp. 6–28, Quarter 2008.

[13] M. A. S. Jr., P. S. Barreto, C. B. Margi, and T. C. Carvalho, "A survey on key management mechanisms for distributed wireless sensor networks," *Computer Networks*, vol. 54, no. 15, pp. 2591 – 2612, 2010.

[14] J. Zhang and V. Varadharajan, "Wireless sensor network key management survey and taxonomy," *Journal of Network and Computer Applications*, vol. 33, no. 2, pp. 63–75, 2010.

[15] H. Moosavi and F. Bui, "A game-theoretic framework for robust optimal intrusion detection in wireless sensor networks," *Information Forensics and Security, IEEE Transactions on*, vol. 9, no. 9, pp. 1367–1379, Sept 2014.

[16] S. Shamshirband, N. B. Anuar, M. L. M. Kiah, V. A. Rohani, D. Petkovi, S. Misra, and A. N. Khan, "Co-FAIS: Cooperative fuzzy artificial immune system for detecting intrusion in wireless sensor networks," *Journal of Network and Computer Applications*, vol. 42, no. 0, pp. 102 – 117, 2014.

[17] M. A. Simplício-Jr., P. S. Barreto, C. B. Margi, and T. C. Carvalho, "A survey on key management mechanisms for distributed wireless sensor networks," *Computer Networks*, vol. 54, no. 15, pp. 2591 – 2612, 2010.

[18] 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.

[19] J. Deng, C. Hartung, R. Han, and S. Mishra, "A practical study of transitory master key establishment forwireless sensor networks," in *SECURECOMM '05: First International Conference on Security and Privacy for Emerging Areas in Communications Networks*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 289–302.

[20] C. Hartung, J. Balasalle, and R. Han, "Node compromise in sensor networks: The need for secure systems," Department of Computer Science University of Colorado at Boulder, Tech. Rep. CU-CS-990-05, 2005.

[21] C. Celozzi, F. Gandino, and M. Rebaudengo, "Hierarchical key negotiation technique for transitory master key schemes in wireless sensor networks," in *Broadband and Wireless Computing, Communication and Applications (BWCCA), 2013 8th International Conference on*, Oct 2013, pp. 151–157.

[22] J. C. Lee, V. C. Leung, K. H. Wong, J. Cao, and H. C. Chan, "Key management issues in wireless sensor networks: current proposals and future developments," *IEEE Wireless Comm.*, vol. 14, no. 5, pp. 76–84, October 2007.

[23] L. Eschenauer and V. Gligor, "A key-management scheme for distributed sensor networks," in *Computer and communications security: CCS '02, 9th ACM conf. on*, 2002, pp. 41–47.

[24] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Symposium on Security and Privacy.*, May 2003, pp. 197–213.

Filippo Gandino obtained his M.S. in 2005 and Ph.D. degree in Computer Engineering in 2010, from the Politecnico di Torino. He is currently an Assistant Professor with the Dipartimento di Automatica e Informatica, Politecnico di Torino. His research interests include ubiquitous computing, RFID, WSNs, security and privacy and network modeling.

Renato Ferrero received the M.S. degree in computer engineering in 2004 and the Ph.D. degree in Computer Engineering in 2012, both from Politecnico di Torino, Italy. He is currently a research fellow at the Dipartimento di Automatica e Informatica of Politecnico di Torino. His research interests include ubiquitous computing, wireless sensor networks and RFID systems.

Bartolomeo Montrucchio received the M.S. degree in Electronic Engineering and the Ph.D. degree in Computer Engineering both from Politecnico di Torino, Italy, in 1998, and 2002, respectively. He is currently an Associate Professor of Computer Engineering at the Dipartimento di Automatica e Informatica of Politecnico di Torino, Italy. His current research interests include image analysis and synthesis techniques, scientific visualization, sensor networks and RFIDs.

Maurizio Rebaudengo received the MS degree in Electronics (1991), and the PhD degree in Computer Engineering (1995), both from Politecnico di Torino, Italy. Currently, he is an Associate Professor at the Dipartimento di Automatica e Informatica of the same institution. His research interests include ubiquitous computing and testing and dependability analysis of computer-based systems.