RoK: A Robust Key Pre-distribution Protocol for Multi-Phase Wireless Sensor Networks

Claude Castelluccia
INRIA - Rhône-Alpes
Inovallée - 655 avenue de l'Europe
38 330 Montbonnot Saint Ismier, FRANCE
email: *claude.castelluccia@inrialpes.fr*

Angelo Spognardi
Dipartimento di Informatica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 - Roma, ITALY
email: *spognardi@di.uniroma1.it*

***Abstract*—Wireless sensor networks are usually deployed to operate for a long period of time. Because nodes are battery-operated, they eventually run out of power and new nodes need to be periodically deployed to assure network connectivity. This type of networks is referred to as Multi-phase WSN in the literature [1]. Current key pre-distribution schemes, such as [2] and [3], are not adapted to multi-stage WSN. With these schemes, the security of the WSN degrades with time, since the proportion of corrupted links gradually increases. In this paper, we propose a new pre-distribution scheme adapted to multi-phase WSN. In the proposed scheme, the pre-distributed keys have limited lifetimes and are refreshed periodically. As a result, a network that is temporarily attacked (i.e. the attacker is active only during a limited amount of time) automatically self-heals, i.e. recovers its initial state when the attack stops. In contrast, with existing schemes, an attacker that corrupts a certain amount of nodes compromises a given fraction of the total number of secure channels. This ratio remains constant until the end of the network, even if the attacker stops its action.**

Furthermore, with our scheme, a network that is constantly attacked (i.e. the attacker regularly corrupts nodes of the network, without stopping) is much less impacted than a network that uses existing key pre-distribution protocols. With these schemes, the number of compromised links constantly increases until all the links are compromised. With our proposal, the proportion of compromised links is limited and constant.

I. Motivations and contributions

Motivation: Key management is a core mechanism to secure wireless sensor networks. The goal of key management is to establish secret keys between sensor nodes that need to communicate securely. One important characteristics of key management protocols is that they must be scalable, CPU and energy efficient. In this paper, we are considering large networks of battery-operated wireless sensors. We are assuming that the average lifetime of each node is much shorter that the operating lifetime of the overall network. As a result, new nodes are periodically deployed in order to assure network connectivity. Each set of new nodes that join the network in a future time consist of a *node generation*. Of course, new nodes must be able to establish secret keys with previously deployed nodes. Protocols that provide this property are known as *Multi-phase* deployment protocols.

Public key cryptography cost is prohibitive for most WSNs and can rarely be used. Most existing key management schemes are based on symmetric key cryptography. One of the most popular schemes, referred in this paper as *RKP* (Random Key Pre-Distribution), was proposed by Eschenauer and Gligor [2] and later on extended by Chan, Perrig and Song [3]. This scheme is distributed and uses a random key pre-distribution approach. In this protocol, each node is configured with a key ring of *m* sub-keys. These keys are randomly drawn from a large key pool of *P* sub-keys. Two nodes establish their secret key from the sub-keys they have in common in their key ring. If the parameters *m* and *P* are chosen properly, the probability that any two nodes share at least one common sub-key is high. This protocol is scalable, CPU and energy efficient. It also trivially supports node addition: any new deployed node gets configured with m sub-keys from the system key pool. It can then establish a secret key with any previously deployed node.

One important drawback of this key pre-distribution scheme is that an attacker that corrupts several nodes can partially reconstruct, from the compromised nodes key rings, the key pool of system. The more nodes it corrupts the more sub-keys it obtains and the more communications it can eavesdrop. If the attacker is constantly corrupting nodes, it will eventually learn the whole key pool and all newly deployed nodes will establish links that will immediately be compromised. In other words, *the security of the whole network degrades with time*. We believe this is a non-desirable property. A naive solution would be to periodically refresh the key pool, i.e. configure new deployed nodes with fresh sub-keys. Newly deployed nodes would be more secure (since their sub-keys were not exposed previously) but would be unable to establish secure links with previously deployed nodes. Newly deployed nodes would constitute a new network that is unable to securely communicate with the previous one. A novel approach is definitely needed.

Contributions: This paper presents a new key pre-distribution scheme, referred as *RoK* (A Robust Key Pre-distribution Protocol for Multi-Phase Wireless Sensor Networks) in the rest of this paper, that allows sensors deployed at different times to establish secure links. In this scheme, sub-keys have limited lifetimes and are refreshed periodically. This has the two following positive consequences:

1) A network that is temporarily attacked (i.e. the attacker is active only during a limited amount of time) automatically **self-heals**, i.e. recovers its initial state when

the attack stops. With the *RKP* scheme, an attacker that corrupts a certain amount of nodes compromises a given percentage of the total number of secure communications. This percentage remains constant until the end of the network. With our proposal, the percentage of compromised links gradually decreases to 0 when the attack stops. *The security of the network improves with time.*

2) A network that is constantly attacked (i.e. the attacker regularly corrupts nodes of the network, without stopping) is much less impacted than a network that uses the basic key pre-distribution protocol. With the *RKP* scheme, the fraction of compromised links constantly increases until it eventually reaches 1 i.e. until all the links are compromised. With our extension, the proportion of compromised links is limited and constant. *The security of the network is constant and does not degrade with time*.

Organization: The next section describes, in detail, our proposal. Its explains how nodes get configured and how they establish secure channels. Section III evaluates the security of our scheme and compares it with the security of the RKP scheme. This evaluation is performed by simulations and analytically. It also discusses some issues such as counterfeiting protection. Related work is summarized in Section IV and Section V concludes this paper.

II. A Robust Key Pre-distribution Protocol for Multi-Phase Wireless Sensor Networks

A. Overview

Because a sensor is battery-powered, it has a limited lifetime. Very often, the sensor's lifetime is much smaller than the lifetime of the whole network. In order to assure a good network connectivity, new sensors need to be deployed periodically to make up for disappearing ones.

In our scheme, we assume that new sensors are deployed at regular epoch that we call *generations*. The time between two consecutive generations is called *generation period*. We also assume that a sensor lifetime is bounded by a given number, Gw, of generations. We refer to Gw as the *generation window*. In other words, a sensor deployed at generation j will run out of power before generation $j+Gw$. For simplicity, we assume in the rest of this paper that the generation period is the time unit, i.e. that the generation period is set to 1.

Newly deployed nodes should be able to establish secure links with previously deployed nodes otherwise connectivity will not be provided. In our scheme, a node deployed at generation j can establish a secret channel with any other sensor deployed in the time period $[j-k, j+k]$, where k is a integer. Note that we assume, for simplicity, in this paper that $k = Gw$. Therefore, since all nodes deployed before generation $j - Gw$ have died, our newly-deployed sensor can establish a key with any other node of the network. Note that, in some scenario, it might make sense to choose a value of k that is smaller then Gw. In this case, the newly deployed node will only be able to establish a secure channel with a subset of the network sensors.

Our scheme is similar to the key pre-distribution scheme described by Eschenauer and Gligor [2]. However as opposed to this scheme, we assign to each sensor A two different key rings: FKR_A and BKR_A, where a key ring is a subset of a pool of keys, namely $FKR_A \subset FKP$ and $BKR_A \subset BKP$.

These key rings are respectively called the *forward* and the *backward* key ring. Similarly, the key pool are called the *forward* and the *backward* key pool. The pools of keys are refreshed at every generation.

The rest of this section describes how the two key pools *FKP* and *BKP* are built and refreshed, how the key rings are assigned to nodes and how nodes establish secret keys.

Table I
Summary of Notation

Gw	generation window
n	last generation of the network
A	sensor A
$kr_A^j = (FKR_A^j, BKR_A^j)$	key ring of A at gen. j
FKR_A^j	Forward key ring of A at gen. j
BKR_A^j	Forward key ring of A at gen. j
m	key ring size
FKP^j	Forward key pool at gen. j
BKP^j	Backward key pool at gen. j
P	key pool size
$fk_t^j \in FKP^j$	t-th $f$key at gen. j
$bk_t^j \in BKP^j$	t-th $b$key at gen. j
h	secure hash function
	$h : \{0,1\}^* \rightarrow \{0,1\}^{160}$
f	hash function
	$f : \{0,1\}^* \rightarrow \{0,1\}^{log_2(P)}$
RKP	key management defined in [2]
RoK	our scheme

B. Key Pools Generation

In the *RKP* scheme [2], the key pool is composed of random keys that do not evolve with time. In contrast, in our scheme, the key pools evolve with time: they are updated at each generation.

The Forward key pool: The forward key pool is initiated with $P/2$ random keys. At each generation, each key is updated by hashing the current key with a secure hash function, $h : \{0,1\}^* \rightarrow \{0,1\}^{160}$, such as SHA1 [4].

More precisely, the forward key pool at generation 0 (i.e. when the network is first deployed) is defined as follows:

$$FKP^0 = \{fk_1^0, fk_2^0, \ldots, fk_{P/2}^0\}$$

where each fk_i^0 is randomly generated. We call $f$keys the keys of the forward key pool.

At generation $j+1$, the $f$keys are refreshed as follows:

$$FKP^{j+1} = \{fk_1^{j+1}, fk_2^{j+1}, \ldots, fk_{P/2}^{j+1}\}$$

where $fk_t^{j+1} = h(fk_t^j)$, and

$$FKP^j = \{fk_1^j, fk_2^j, \ldots, fk_{P/2}^j\}$$

defines the forward key pool at generation j.

The Backward key pool: The backward key pools, composed of backward keys, are generated using a hash-based Lamport chain [5]. If we assume that the network is deployed at generation 0 and will last, at most, until generation n, we start by generating the key pool of generation n. The $b$keys (backward keys) at generation n, i.e. the last generation of the network, are initialized with random values i.e.:

$$BKP^n = \{bk_1^n, bk_2^n, \ldots, bk_{P/2}^n\}.$$

At generation j, the $b$keys are refreshed as follows:

$$BKP^j = \{bk_1^j, bk_2^j, \ldots, bk_{P/2}^j\}$$

where $bk_t^j = h(bk_t^{j+1})$ and

$$BKP^{j+1} = \{bk_1^{j+1}, bk_2^{j+1}, \ldots, bk_{P/2}^{j+1}\}$$

defines the backward key pool at generation $j+1$.

That is, the bk_t^j key of generation j is obtained from the key bk_t^{j+1} of generation $j+1$, using the hash function h.

C. Key rings assignment

This section describes how the backward and forward key rings are assigned to each sensor.

Each node is configured with $m/2$ subkeys from the backward and forward key pools. The subkeys assigned to a given node A deployed at generation j are selected using a pseudo-random function. More specifically, the first subkey of the forward key ring will be the subkey with index $f(id_A||1||j)$ of the forward key pool, where $f(.)$ is for example a hash function with range $[1; m]$. Similarly, the first subkey of the backward key ring will be the subkey with index $f(id_A||1||j)$ of the backward key pool. The second subkey of the forward key ring will be the subkey with index $f(id_A||2||j)$ of the forward key pool. Similarly, the second subkey of the backward key ring will be the subkey with index $f(id_A||2||j)$ of the backward key pool and so on.

More formally, node A is configured with a key ring, $kr_A^j = (FKR_A^j, BKR_A^j)$, defined as follows:

$$FKR_A^j = \{fk_t^j \mid t = f(id_A||i||j),\ i = 1, 2, \ldots, m/2\}$$

$$BKR_A^j = \{bk_t^{j+Gw-1} \mid t = f(id_A||i||j),\ i = 1, 2, \ldots, m/2\}$$

and m is the size of the whole key ring.

Note that all the key rings are strictly bound to the deployment generation of the owner sensor. As a result, nodes need to be loosely time-synchronized.

Note that node A can only update its key ring kr_A^i, for the generations i between j and $j+Gw$. In fact since it cannot compute the $b$keys for the generations after generation $j+Gw$, it cannot update kr_A beyond generation $j+Gw$. Furthermore, since it cannot recover any $f$keys for the generations before generation j, it cannot compute kr_A for the generation before j. As a result, the lifetime of its key ring is limited. As we will see later, this is an essential aspect of our scheme since it considerably limits the power of the attacker: an attacker that corrupts a node will only be able to use its keys for a limited period of time.

D. Establishing a secure channel

When a sensor A is deployed at generation j, it initiates the neighbors discovery procedure by broadcasting a message that includes its identifier, id_A, and its generation.

All the nodes that receive its message are able to reconstruct the list of the key indexes in kr_A and identify the keys that they have in common.

For example, a neighbor node B can construct the set $\{t \mid t = f(id_A||l||j),\ l = 1, 2, \ldots, m/2\}$ of the key indexes and check if it shares at least one index t with A. The B node then replies with its identifier, id_B, and its generation. As a result, A identifies the list $t_1, t_2, \ldots$ of all the key indexes in common.

Both A and B then calculate their *overlapping generations*, that is the set of the generations in which they can communicate. Formally, if A is deployed at generation j and B was deployed at generation i with $i \leq j$, their overlapping generations will be all the generations in the interval $[j, i+Gw[$.

Notice that if A and B have in common the key indexes $t_1, t_2, \ldots, t_z$, then both of them compute all the $f$keys $\{fk_\tau^\gamma | \tau = t_1, t_2, \ldots, t_z, \gamma = j, \ldots, i+Gw-1\}$ and all the $b$keys $\{bk_\tau^\gamma | \tau = t_1, t_2, \ldots, t_z, \gamma = j, \ldots, i+Gw-1\}$.

A and B can then compute their secret key as follows:

$$k_{AB} = h(fk_{t_1}^j || \quad bk_{t_1}^{i+Gw-1} || fk_{t_2}^j || bk_{t_2}^{i+Gw-1} || \ldots \\ \ldots || fk_{t_z}^j || bk_{t_z}^{i+Gw-1})$$

In this formula, the forward keys fk_x^j provide forward security: The forward keys of corrupted nodes, that happen to have the same key indexes that those used to compute k_{AB}, are only useful if they are corrupted before generation j. The backward keys bk_x^{i+Gw-1} provide backward security: The backward keys of corrupted nodes, that happen to have the same key indexes that those used to compute k_{AB}, are only useful if they are corrupt after generation i.

The key k_{AB} can then be used to established a secure channel between sensors A and B.

At each new generation, each node should erase their forward keys of the previous generation. By doing this, the $f$keys of the previous generations not yet retrieved by the adversary cannot be compromised any more: all nodes will have the $f$keys of the new generation and none of the nodes that can be captured will maintain a copy of the previous $f$keys. As a result, all the secure channels established with $f$keys of previous generation j cannot be compromised anymore. As we will see in Section III, this significantly increases security.

E. Example

This section illustrates our protocol with an example (see Figure 1). Let's assume two sensors A and B, deployed respectively at generation j and $j+2$, with a generation window Gw of 4. The overlapping generations are therefore $j+2, j+3$ and $j+4$. Let's also assume that A and B share the keys t_1 and t_2 (that is $t_\tau = f(A||l_{A_\tau}||j) = f(B||l_{B_\tau}||j+2)$ for some l_{A_τ} and l_{B_τ}, with $\tau = 1, 2$).

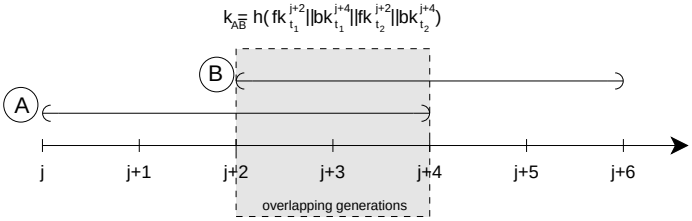

Figure 1. Key establishment: A and B share the keys t_1 and t_2

A and B can then compute the common secret key

$$k_{AB} = h(fk_{t_1}^{j+2}||bk_{t_1}^{j+4}||fk_{t_2}^{j+2}||bk_{t_2}^{j+4}).$$

When generation $j+3$ arrives, A and B update their forward key ring by hashing all their $f$keys. They must also erase from memory their forward keys of the generation $j+2$. Note that the keys used by node A are only valid between generation j and $j+4$. Similarly, the keys used by node B are only valid between generation $j+2$ and $j+6$.

III. Security Evaluation

A. Objectives

Our scheme improves the security of the *RKP* scheme by limiting the lifetime of the key pools and refreshing the pool subkeys. With *RKP*, an attacker that corrupts enough sensors can reconstruct a large part of key pool and compromise many of the established or upcoming links. Since the key pool does not change, newly deployed nodes are configured with some corrupted subkeys. In other words, the security degrades with time.

In contrast, in our scheme, the key pools evolve with time. As a result, since newly deployed nodes are configured with fresh subkeys, an attacker has to constantly corrupt nodes and be very aggressive if he wants to eavesdrop on the established communications. If the attacker operates only during a limited period of time, the network will automatically self-heal when the attacker stops compromising nodes.

The goal of this section is to evaluate the security of our proposal and compare it with the security of the *RKP* scheme.

We evaluate the security of these two schemes by evaluating the number of channels that get *indirectly* corrupted when x nodes are compromised (and their keys are disclosed). A channel, between nodes A and B, is said to be indirectly corrupted when neither A nor B have been corrupted, but the adversary has collected all the backward and forward sub-keys that A and B have in common. These sub-keys have been collected by compromising other nodes.

We evaluate the two following ratios:

1) R_{active} is the ratio of the number of *indirectly* corrupted *active* channels over the total number of active channels. A channel is active when both of its ends are still alive. An attacker that corrupts an active channel can decrypt all the messages that are sent of the link, and can also inject bogus messages by impersonating one of the communication nodes.
2) R_{total} is the ratio of the total number of *indirectly* corrupted channels over the total number since the beginning of the network, over the total number of channels that have been established since the birth of the network. An attacker that corrupts a non-active channel, can decrypt the messages that were sent over these channels, if these messages were recorded. However, these messages are old and might not be valuable anymore. Furthermore, an attacker that corrupts a non-active node cannot inject bogus messages, since the secret key it has recovered has expired and cannot be used anymore to send new messages. For these reasons, we believe the R_{active} is more important. However, for completeness, we evaluate both ratios.

We first evaluate these ratios for both schemes, *RKP* and *RoK*, by simulation. This allows us to consider elaborate and complex scenarios. As detailed in the following section, we consider different types of attackers. We then compute analytically the ratio R_{active}. We believe this is also useful for a network designer to understand and control the security of its network. He can, for example, compute for some sets of parameters, the security of its network (i.e. fraction of corrupted links). Alternatively, he can set the maximum acceptable fraction of corrupted links, and use our formulas to derive the network parameters (i.e., for example, the sensor node lifetime).

B. Evaluation by Simulation

1) Simulation Set-up: We have simulated our scheme, referred to as *RoK*, and the Eschenauer and Gligor scheme, referred to as *RKP*.

To simplify the security analysis, we modeled the network as a grid of sensors of size 400. We assumed that the number of neighbors, i.e. of nodes in the communication range, of each sensor is constant and equal to four.

When nodes are deployed, they establish a secure channel with their four neighbors, using all the sub-keys they share. If the intersection of the key rings of two neighboring nodes, A and B, is empty, node A (resp. B) checks whether any of its 2-hop neighbors shares at least a subkey with B (resp. A). If this is the case, this 2-hop neighbor is used as a proxy. If not, A extends its search to its 3-hop neighbors. This algorithm proceeds until a node that shares a subkey with B (resp.A) is found. This node is then used as a proxy.

In order to be fair, we used the same memory requirement for both schemes. The *RKP* scheme used a key pool of size P and each node was configured with a key ring size of m. In contrast, the *RoK* scheme used a forward key pool of size $P/2$, a backward key pool of size $P/2$ and each node was configured with two key rings, each of size $m/2$. Since the connectivity (the probability of two neighbors to establish a key) depends on the size of the key pool, the connectivity performance of our scheme, that uses a key pool of size $m/2$, is weaker than the connectivity achieved with the *RKP* scheme. However since our scheme uses proxies, the overall network connectivity achieved with these two schemes are similar. In

our simulations, P was set to 20000 and m to 500. These parameters are similar to the ones used in [2] and [3].

We assumed that each generation is composed of 10 time slots. At generation 0, N nodes are deployed. We simulated nodes expiration by assigning to each node a random expiration date, chosen according to a Gaussian distribution with mean $\frac{Gw}{2}$ and with standard deviation $\frac{Gw}{6}$. Where not differently specified, we used a generation window $Gw = 10$. In order to simplify the security analysis, we assumed that the network topology does not change over time: At each generation, expired nodes are replaced with new ones, configured with fresh keys. The new nodes establish secure channels with their four neighbors, using the common keys of the least and the greatest overlapping generations (respectively for the $f$keys and for the $b$keys).

2) Attacker model and strategy: It is important to underline that the $f$keys and $b$keys can be computed separately. For a captured node of generation j, $f$keys with the same subscripts of all future generations after j (even after $j+Gw$) and $b$keys with the same subscripts of all generations before $j+Gw$ can easily be computed, stored and exchanged among captured nodes. In this way, the attacker may create a table of keys that belong to various generations. This table of keys might be used to compromise some additional secure links of various generations (including the past ones).

In our simulations, the attacker uses the strategy described above: at each time slot, he corrupts x active nodes and updates the previously described table with their backward and forward keys. He then uses this table to corrupt links (alive or past).

We considered two different types of attackers: the *eager*, and the *temporary* attackers. An eager attacker keeps compromising nodes at constant rate, from the deployment of the first generation of sensors to the end of the network. In contrast, temporary attacker compromises nodes during a limited period of time, from generation 5 to generation 14 in our simulations.

We then counted, at each time slot, the number of compromised links (i.e. links that were secured using forward and backward compromised keys that are known to the attacker), and computed the ratios R_{active} and R_{total}, described in Section III-A.

In the next section we report the results of our simulation. All the simulations were repeated 25 times and the results report the average values.

3) Simulation results: This section presents the results of our simulations for the different types of attackers.

Eager Attacker model: Figure 2 and 3 display, respectively, the R_{active} and R_{total} ratios with an eager attacker. These figures present the simulation results for three different corruption rates: 1, 3 and 5 nodes at every time slot, i.e. respectively 10, 30 and 50 nodes per generation. Notice that we only consider *indirectly* corrupted channels, i.e. channels established between non-compromised nodes.

The main observation is that while the *RKP* scheme has ratios of compromised channels with an almost constant

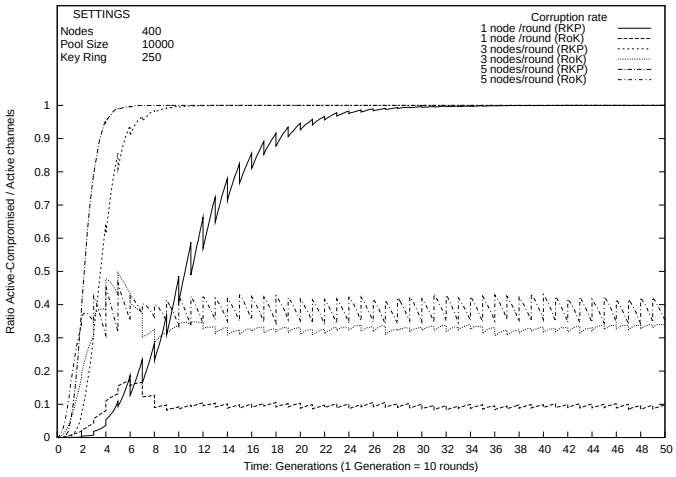

Figure 2. R_{active}. Adversary compromises nodes at constant rate (constant attacker)

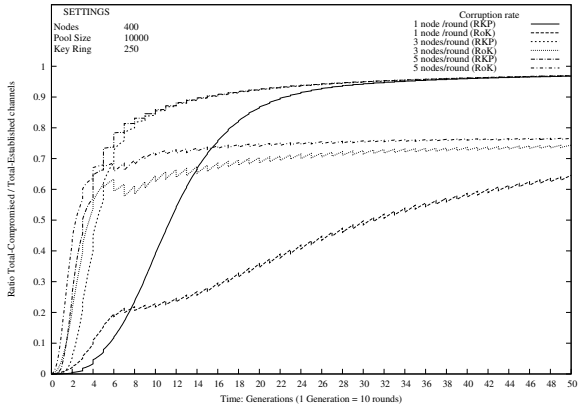

Figure 3. R_{total}. Adversary compromises nodes at constant rate (constant attacker)

growth, the ratios obtained with the *RoK* scheme are bounded by a threshold value, that depends on the value x. Secondly, from Figure 2 it can be observed that with *RKP*, the R_{active} ration reaches 1 in a really short time (around 30 generations for corruption rate 10 and 8 generations for corruption rate 50), while with *RoK* the ratios are much smaller and always below $1/2$.

The oscillations in the plots are due to newly-deployed nodes: the new nodes establish secure channels with keys not yet compromised by the adversary, suddenly reducing the ratio compromised/active channels. In the *RKP* scheme, the oscillation amplitude reduces as the adversary keeps acquiring keys. With the *RoK* scheme the amplitude is almost constant, for the whole life of the network, since keys are refreshed at every generation.

We can also observe that the plots for the *RoK* scheme in Figure 2 have an increasing phase followed by a decreasing one. Furthermore, for all three corruption rates, the plots reach their maximum around the 5-th generation. This apparently

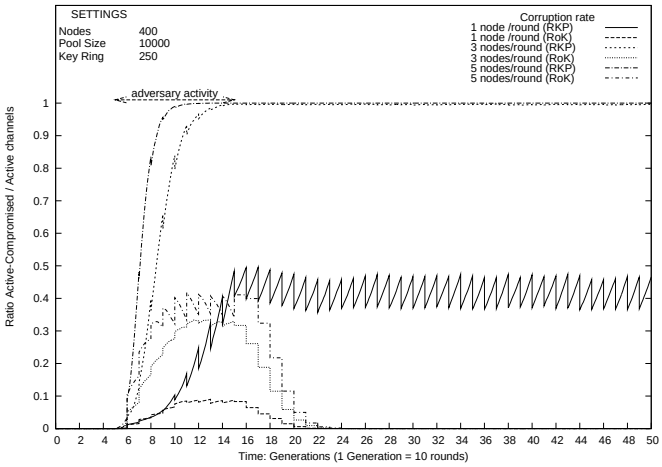

Figure 4. R_{active}. Adversary compromises from generation 5 to generation 15 (temporary attacker)

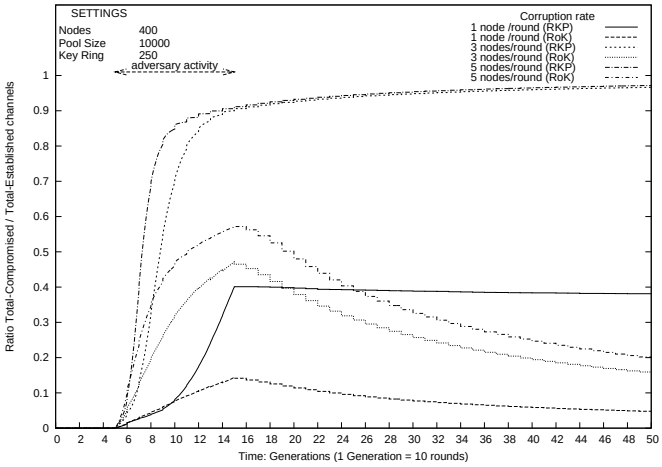

Figure 5. R_{total}. Adversary compromises from generation 5 to generation 15 (temporary attacker)

unexpected behavior is due to the proximity of the deployment time: at generation 5, almost half of the nodes of the network were deployed at the first generation and almost all the keys that the adversary has collected can be used to exploit the channels of those nodes (since $Gw = 10$, the average lifetime of a node is $Gw/2 = 5$). From the 6-th generation on, most of the nodes are "young" and the channels were established with keys that were not compromised yet. As soon as the population of the nodes of the network stabilizes, the ratio of compromised channels also stabilizes.

Note that corruption ratio of the total number of established channels shown in Figure 3 never reaches 1 because of the directly corrupted channels that are not taken into account.

It can be observed in Figure 3 and Figure 4, that *RKP* outperforms our scheme in the first generations (i.e. from generation 0 to 5, when the network is mainly composed of nodes deployed at the first generation). In fact, since the connectivity achieved with our scheme is smaller, each link is secured with less sub-keys than with the *RKP* scheme and can therefore be compromised more easily. However, as new nodes are deployed in the network and new channels are established with fresh sub-keys, the performance of our scheme significantly increases with time.

Temporary Attacker Model: The results for the temporary attacker are collected in Figure 4 and in Figure 5. The interval of action of the attacker (from generation 5 to generation 14) is denoted with the label "adversary activity".

We simulated a network with the same settings as the network used for the eager attacker, and the three corruption rate of 10, 30 and 50 nodes per generation.

Both figures illustrate the *self-healing* property of the proposed scheme: as soon as the adversary stops its activity, the ratio of the compromised channels starts decreasing as new generations of nodes are deployed. Figure 4 in particular shows that the network starts recovering as soon as the attack stops. It takes 10 generations for the network to fully recover.

The *RKP* scheme, instead, keeps a ratio of compromised channels greater than 0, even when the adversary stops its activity. Moreover, when the attacker corrupts nodes with a rate greater than 30, the ratio approximates 1 and never decreases. Figure 5 shows that with *RoK* the ratio R_{total} decreases when the adversary stops; for the *RKP* scheme the ratio keeps increasing toward 1.

In summary, these simulation results show that our *RoK* scheme outperforms the RKP scheme. The number of active compromised nodes can be reduced by a factor of 10. For example, when 10 nodes are compromised at each generation, the ratio of active compromised channels is reduced from 100% to 10%. The security of our proposal is based on the assumption that it takes time for an adversary to physically compromise sensors and get their keys. Since in our proposal, the key vulnerability period, i.e. the time that attacker has to corrupt useful keys is reduced, security is improved significantly.

C. Analytical Evaluation

1) R_{active} computation: In this section, we compute analytically R_{active}, the fraction of active indirectly compromised links when x nodes get compromised. We assume that most of the links are established directly, i.e. each node shares at least one sub-key with each of its neighbors.

As explained in [3], the probability that two nodes share i sub-keys is defined by:

$$p_i = \frac{\binom{P}{i}\binom{P-i}{2(m-i)}\binom{2(m-i)}{(m-i)}}{\binom{P}{m}^2} \quad (1)$$

where m is the key ring size and P the key pool size.

Therefore the probability that two nodes share at least one sub-key is defined by $1 - p_0$ i.e:

$$pconnect = 1 - \frac{\binom{P}{2m}\binom{2m}{m}}{\binom{P}{m}^2} \quad (2)$$

We need to choose m and P such that $pconnect \sim 1$, i.e., for example, $m = 250$ and $P = 10000$.

Furthermore, a secure link will be created by, on average, nk sub-keys, where nk is defined as follows:

$$nk = \sum_{i=1}^{+\infty} p_i \cdot i \tag{3}$$

Note that for $m = 250$ and $P = 10000$, $nk \sim 6$.

We assume in our scheme that n nodes are deployed at bootstrap time (generation 0, G_0). Nodes that die during generation G_i are immediately replaced at generation G_{i+1}. The time between two deployments is called generation *period*.

We also assume that each sensor has a lifetime that follows a distribution, for example Gaussian, with a density probability $f(x, \mu, \sigma)$, where μ defines the mean lifetime (the unit is the generation period) and σ the standard deviation. The density probability $f(x, 3, 1/6)$ indicate that the mean lifetime of a sensor is 3 generations and its standard deviation is $1/6$ generation. The lifetime of a node is bounded by Gw, i.e if a node is deployed at generation G_j, it will, for sure, be dead after generation $G_j + Gw$.

It can easily be shown that the number of nodes that died at generation G_{j-1}, and redeployed at generation G_j, with $j > 1$, is defined by:

$$X_j = \sum_{i=1}^{Gw} X_{j-i} \int_{i-1}^{i} f(x, \mu, \sigma)\ dx \tag{4}$$

with $X_0 = n$.

If the sensor lifetime follows a Gaussian distribution $f(x, \mu, \sigma)$, we have:

$$X_j = \sum_{i=1}^{Gw} X_{j-i} \cdot (F(i; \mu, \sigma) - F(i-1; \mu, \sigma)) \tag{5}$$

where

$$F(x; \mu, \sigma) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]. \tag{6}$$

We can demonstrate that, for j enough large, X_j becomes constant: the network stabilizes and, then, the average number of nodes to re-deploy is constant.

If we denote, $X^{(i)}$ the number of active nodes deployed i generations ago, we can have:

$$X^{(i)} = X_j \int_{i}^{Gw} f(x, \mu, \sigma)\ dx \tag{7}$$

$$= X_j(F(Gw; \mu, \sigma) - F(i; \mu, \sigma)) \tag{8}$$

From Formula 8 we can compute that, on the average, a node picked at random from the network has age i, with $0 < i < Gw$, with probability:

$$p(i) = \frac{1}{n} \cdot X^{(i)} \tag{9}$$

The average age $E[\alpha]$ of nodes is therefore defined as:

$$E[\alpha] = \sum_{i=0}^{Gw} i \cdot \frac{1}{n} X^{(i)} = \frac{1}{n} \sum_{i=0}^{Gw} i X^{(i)} \tag{10}$$

Let the average number of captured nodes be x during a period. Since each node contains m keys, the probability that a given key has not been compromised is $(1 - \frac{m}{P})^x$. Therefore if a given link was secured with i sub-keys, the probability that this link is compromised is $(1 - (1 - \frac{m}{P})^x)^i$.

According to [3], we defined the average probability that a link is **indirectly**[1] corrupted when x nodes are corrupted as:

$$plc = \sum_{i=1}^{m} \left(1 - \left(1 - \frac{m}{P}\right)^x\right)^i \tag{11}$$

Therefore, with the ***RKP*** scheme, the probability that an active link is compromised at generation j is defined as follows:

$$plc_{RKP}(j) = \sum_{i=1}^{m} \left(1 - \left(1 - \frac{m}{P}\right)^{j \cdot x}\right)^i p_i \tag{12}$$

with p_i defined by Formula 1. Note that this probability increases with time.

With ***RoK***, the probability that an active link is compromised at generation G_j depends on the generation of the "oldest" node that established the channel. So, we have to evaluate the average age of the oldest between two random nodes of the network. Let be X and Y two independent random variables with the same distribution defined by Formula 8. Let be $Z = MAX(X, Y)$. The expected value of Z is the average time at disposal of the adversary to indirectly corrupt a random active channel.

The probability that Z assumes the value j, where $0 < j < Gw$ is defined by:

$$\begin{aligned} P\{Z = j\} =& P\{(X = j \wedge Y \leq j) \\ & \vee (X \leq j - 1 \wedge Y = j)\} \end{aligned} \tag{13}$$

$$= p(j) \cdot \sum_{k=0}^{j} p(k) + \sum_{k=0}^{j-1} p(k) \cdot p(j) \tag{14}$$

$$= p(j)^2 + 2\left(p(j) \cdot \sum_{k=0}^{j-1} p(k)\right) \tag{15}$$

Therefore, $E[Z]$ can be computed as follows:

$$E[Z] = \sum_{j=0}^{Gw} j \cdot P\{Z = j\} = \tag{16}$$

$$= \sum_{j=0}^{Gw} j \left[p(j)^2 + 2\left(p(j) \cdot \sum_{k=0}^{j-1} p(k)\right)\right] \tag{17}$$

Finally, the probability that an active link is compromised at generation j with our scheme is defined as:

$$plc_{RoK}(j) \sim \sum_{i=1}^{m} \left(1 - \left(1 - \frac{m}{P}\right)^{x \cdot E[Z]}\right)^i p_i \tag{18}$$

Note that this probability is constant, since it does not depend on j.

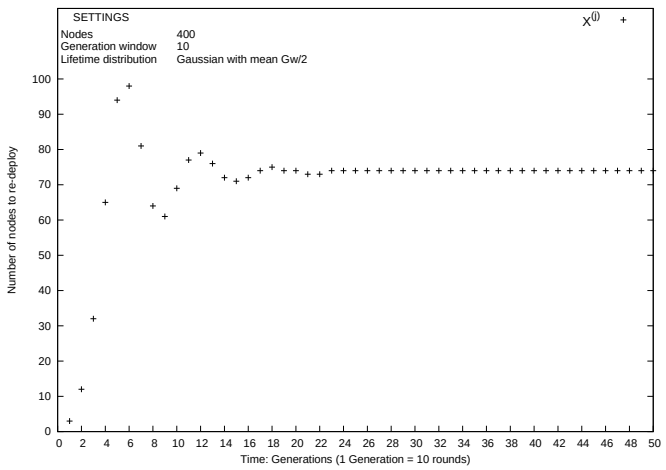

Figure 6. Average number of nodes to re-deploy every generation. For every generation is reported the expected number of nodes died in that generation.

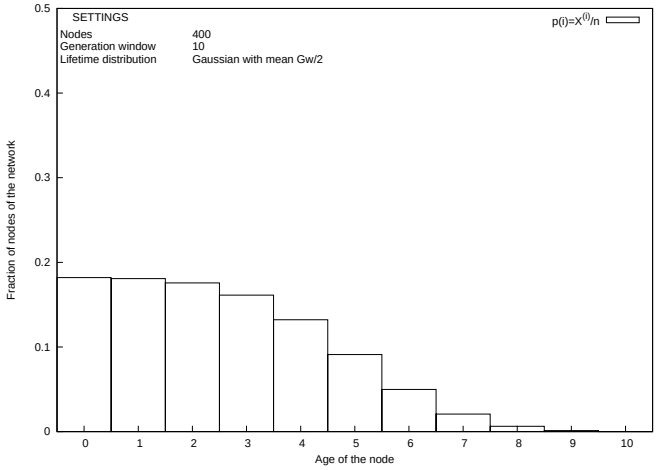

Figure 7. Average age of the nodes. For every age is reported the fraction of nodes with that age.

2) Some numerical results: If $n = 400$, $m = 250$, $P = 10000$, $Gw = 10$, $x = 10$, $ord = 4$, the total number of links is around 800. Evaluating Formula 3, we obtain that $nk \sim 6$.

Formula 5 shows that the average number of nodes to re-deploy every generation converges. This function is plotted in Figure 6. With the previous settings, the function converges to the value 72.

The distribution of the age of a node can be evaluated via Formula 9, plotted in Figure 7.

The expected age (Formula 10) is 2.5, while the expected age of the oldest between two random nodes (Formula 17) is 3.6. Using Formulas 12 and 18, we obtain that $plc_{RKP}(15) \sim 0.75$, $plc_{RKP}(30) \sim 0.993$ and $plc_{RKP}(50) \sim 0.9999$. Furthermore, $plc_{RoK}(10, 30, 50) \sim 0.10$ (and 83 links are directly compromised). These results demonstrate that the *RoK* scheme reduces the numbers of compromised channels by a factor of 10. They also show that, with *RoK*, the ratio of compromised channels remain constant with time. In contrast, the number of compromised channels increases with *RKP*, i.e. security degrades with time.

Figure 8 compares the results obtained analytically with formulas 12 and 18 with the results obtained by simulations. This figure demonstrates that the results obtained analytically and by simulations are very similar. Note that in the simulations, it is assumed that the attacker corrupts nodes regularly within a generation. In the analytical model, we assume, for simplicity, that the attacker corrupts all the nodes at once i.e. at the beginning of each generation. This explains why the oscillations that are visible in the simulation results do not appear in the analytical results.

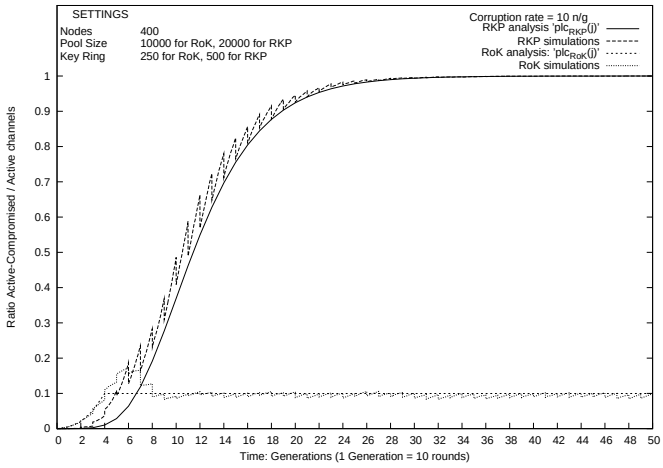

Figure 8. Comparison analysis and simulation results

D. Discussions

1) Counterfeiting protection: Our scheme not only improves the security of deployed nodes but also makes node counterfeiting very difficult. Counterfeiting happens when an attacker generates a new node from the sub-keys it has compromised. With the *RKP* scheme, the attacker has to find an id such that it has compromised all the corresponding sub-keys (or at least the sub-keys it has in common with the nodes it wants to establish a secret channel with). If the attacker has compromised enough nodes, it will know many sub-keys and this operation might not be too difficult.

With our scheme, since sub-keys have limited lifetimes, node generation is much more difficult. As explained in section II-D, two sensors A and B, deployed respectively at generation i and generation j, compute their secret key as follows:

$$k_{AB} = h(fk^j_{t_1} || \quad bk^{i+Gw-1}_{t_1} || fk^j_{t_2} || bk^{i+Gw-1}_{t_2} || \dots \\ \dots || fk^j_{t_z} || bk^{i+Gw-1}_{t_z})$$

If the attacker wants to generate a new node X, let's say deployed at generation x, where $x < j$, in order to establish a secure link with B, it can only use the nodes corrupted in $j - x$ generations (i.e. from generation x to generation j) to

[1] Remember that a link is indirectly corrupted when none of its end-points have been corrupted, but the adversary collected all the keys used to establish the link, recovering those keys form compromised nodes

perform his attack. The lifetime of the established link will then be $Gw-(j-x)$. There is a clear trade-off here between the time at disposal of the attacker to perform his attack and the benefit it gains (i.e. the lifetime of the resulting established channels).

If $j-x$ is small, then the attacker has very limited time to collect the correct sub-keys, but the lifetime of the link might be large i.e. $Gw-(j-x)$. However if the attacker needs more time, i.e. if it decreases x, the duration of the established lifetime will be smaller. We believe that this is a nice feature of our scheme. An attacker needs to be very aggressive and corrupts many nodes very quickly if it wants to benefit from its attacks. Performing such aggressive attacks might not always be possible and requires very strong attackers.

2) Controlling the Security of a WSN: Formula 18 clearly shows that the security provided by our scheme depends on the value of Gw, or in order words, the refresh period of the sub-keys. This formula can be used to evaluate the security of a WSN that has predefined parameters. On the other hand, it can also be used to specify the network parameter Gw according to the administrator security goals and the attacker model. For example, the above example shows that if the attacker can corrupt up to 10 nodes per period (period being the unit of time), the network is composed of 400 nodes, each configured with 250 sub-keys out of 10000 sub-keys, and that the security goal is to make sure that less than 33% of the network is corrupted, then Gw must be set to 10 periods. In order words, the maximum lifetime of each node must be set to 10 periods (and the average lifetime will be 5) and new nodes must be regularly deployed in order to replace dying nodes. Similarly, if the security goal is less than 20% of corruption, Gw must be set to 5 periods. Equation 5 shows that, with the previous parameters, ~ 80 nodes die at every period and need to be replaced. An idea of the impact of the parameter Gw can be had from the Figures 9 and 10, where we compared the ratios introduced in Section III-A, for a fixed and constant rate of corruption and varying Gw.

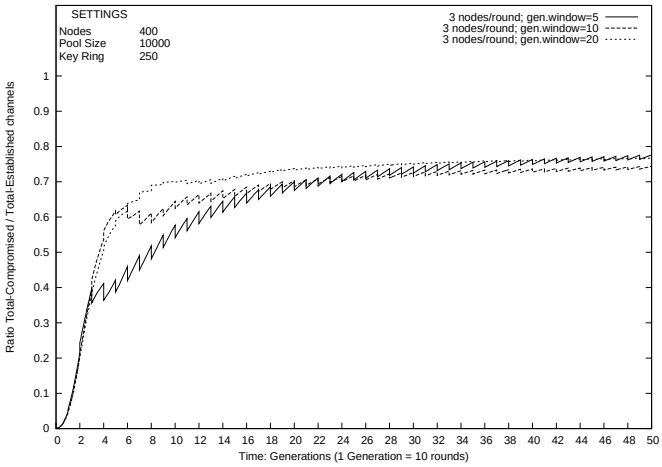

Figure 9. Security of *RoK* with different values of Gw. Ratio between Total-compromised and Total-established channels, for the whole lifetime of the network.

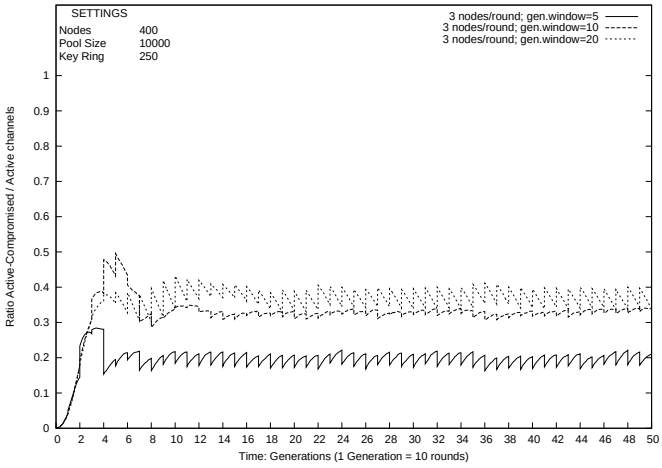

Figure 10. Security of *RoK* with different values of Gw. Ratio between Active-compromised and Active channels.

IV. Related Work

Key management is one of the core mechanism to provide security to WSN. Even though public key based solutions are being investigated in the literature ([6], [7], [8]), these solutions suffer from the CPU, memory and energy limitations of sensors. As a result, most of the existing proposals consider only the use of symmetric key cryptography.

One of the most popular approach, referred as *random key pre-distribution approach*, was proposed by Eschenauer and Gligor [2]. In this scheme, each node is configured with a key ring randomly selected from a larger pool of symmetric keys: they propose a model that relies on probabilistic key sharing among the nodes of a random graph. In [9], Di Pietro et al. provide some new results to study the scheme, providing a new model instead of the traditional one based on the Erdös-Rényi random graphs. The Random Key Pre-distribution Scheme [3] of Chang, Perrig and Song is an extension of [2] and presents three proposals to strengthen security and improve resilience of the established link keys. In [10], [11], a cooperative pair-wise key establishment protocol is proposed, where a link between two nodes is reinforced using the cooperation of the common neighborhood.

While random key pre-distribution schemes are efficient and promising, the security that they provide degrades with time. Our proposal, *RoK*, solves this problem by periodically refreshing the key pool.

One work that takes in account the challenge to protect wireless sensor networks that can be deployed in different phases is [1]. The considered model is very similar to the one considered in this paper: sensors are deployed in successive generations, when previously deployed sensors fail or when the capability of the existing network is determined to be insufficient. All the deployed nodes are able to establish secure channels with nodes of the next n generations. Unlike the previous works, this protocol is not based on a random pre-distribution of the keys. Every node is deployed with exactly

one *generation key* that binds them to a specific generation; for all the other generations it can communicate with, it maintains a secret derived from an own unique random value and the corresponding generation key. Formally, if node A is deployed at generation j, it is configured with a unique random value R_A, the generation key gk_j and a set of secrets $S_{A,j+1}, S_{A,j+2}, \ldots, S_{A,j+n}$, one for every of the n following generations; each secret is constructed as follows: $S_{A,i} = G_{gk_i}(R_A)$, where G is a keyed one-way hash function and gk_i is the generation key of generation i. Clearly sensor A cannot recover gk_i from $S_{A,i}$ and R_A. A node B of generation $j' \leq j+n$, upon receiving R_A will be able to construct $S_{A,j'}$, since it knows the key $gk_{j'}$. Nodes A and B, then, share the secret $S_{A,j'}$ that can use to establish a session key. Since each node only knows the generation key corresponding to its own generation, it is not able to construct the secrets for an arbitrary generation. Moreover, it cannot tamper with the communication between nodes of a different generation, and cannot impersonate a node Z of generation i, since it cannot compute $G_{gk_i}(R_z)$.

The security of the whole protocol is based on the assumption that it takes time for an adversary to physically compromise sensors and get the stored keys. Since the whole security of the scheme relies on the generation key gk_j, all sensors of generation j must erase this key as soon as the key-establishment protocol is over.

The security provided by this protocol is rather weak: since all the nodes of generation j have a copy of gk_j, it is sufficient for the adversary to compromise *one* of those nodes to corrupt all the communications of that generation.

With the *RoK* scheme, discovering the whole traffic of the network is a very difficult task, since the adversary needs to reconstruct the whole key pools. Moreover, *RoK* is more secure since the corruption of a single node has only a very limited impact on the security of the whole network.

V. Conclusion

This paper describes a new key pre-distribution scheme that allows sensors of different generations, i.e. deployed at different times, to establish secure channels. In the proposed scheme, the predistributed keys have limited lifetimes and are refreshed periodically.

We show that a network that is temporarily attacked automatically **self-heals**, i.e. recovers its initial state when the attack stops. In contrast, with existing schemes, an attacker that corrupts a certain amount of nodes compromises a given fraction of the total number of secure channels. This ratio remains constant until the end of the network, even if the attacker stops its action. Furthermore, we show that a *RoK* based network that is constantly attacked is much less affected than a network that uses existing key pre-distribution protocols.

Our analysis and simulation results demonstrate that our *RoK* scheme outperforms the *RKP* scheme proposed by Eschenauer and Gligor [2]. The number of active compromised nodes can be reduced by a factor of 10. The security of our proposal is based on the assumption that it takes time for an adversary to physically compromise sensors and get their keys. Since in our proposal, the key vulnerability period, i.e. the time that attacker has to corrupt useful keys, is reduced, security is improved significantly.

This paper shows that it is possible to control the security of wireless sensor networks by limiting the lifetime of sensors and by deploying new ones periodically. The analytical model, derived in this paper, can be used to compute, according to the security objectives and the attacker model, the network parameters, i.e. the sensor lifetime and re-deployment period.

Acknowledgments

We are grateful to Albert Levi, our shepherd, for his suggestions that helped improve the paper. We would also like to thank the anonymous reviewers for their comments and suggestions.

The work described in this paper is based on results of IST FP6 STREP *UbiSec&Sens* (http://www.ist-ubisecsens.org). *UbiSec&Sens* receives research funding from the European Community's Sixth Framework Programme. Apart from this, the European Commission has no responsibility for the content of this paper. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

References

[1] B. Dutertre, S. Cheung, and J. Levy, "Lightweight key management in wireless sensors networks by leveraging initial trust," SRI International, SDL Technical Report SRI-SDL-04-02, April 2004.

[2] 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, CCS, Washington, DC, USA*, November 2002.

[3] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[4] D. Eastlake 3rd and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," , United States, 2001.

[5] L. Lamport, "Password authentication with insecure communication," *Commun. ACM*, vol. 24, no. 11, 1981.

[6] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz, "Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs." in *CHES: Cryptography Hardware and Embedded Systems*, August 2004.

[7] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *PERCOM '05: Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications*, March 2005.

[8] D. Malan, M. Welsh, and M. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Proceedings of 1st IEEE International Conference on Sensor and Ad Hoc Communications and Network, Santa Clara-CA,USA*, October 2004.

[9] R. Di Pietro, L. V. Mancini, A. Mei, A. Panconesi, and J. Radhakrishnan, "Connectivity properties of secure wireless sensor networks," in *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, SASN, Washington DC, USA*, October 2004.

[10] R. Di Pietro, L. V. Mancini, and A. Mei, "Random key assignment for secure wireless sensor networks," in *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks, SASN, Fairfax-VA, USA*, October 2003.

[11] M. Conti, R. Di Pietro, and L. V. Mancini, "ECCE: Enhanced cooperative channel establishment for secure pair-wise communication in wsn," in *Journal of AdHoc Networks*, Elsevier, Ed., January 2007.