Efficient and Resilient Key Discovery
Based on Pseudo-Random Key Pre-Deployment

Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei
Dipartimento di Informatica
Università di Roma “La Sapienza”
Via Salaria, 113 - 00198 Roma - Italy
{dipietro,mancini,mei}@di.uniroma1.it

Abstract

A distributed Wireless Sensor Network (WSN) is a collection of n sensors with limited hardware resources and multi-hop message exchange capabilities. Due to the scarceness of resources, the distributed paradigm required, and the threats to the security, a challenging problem is how to implement secure pairwise communications among any pair of sensors in a WSN. In particular, storage memory and energy saving as well as resilience to physical compromising of a sensor are the more stringent requirements. The contributions of this paper are twofold: (1) we describe a new threat model to communications confidentiality in WSNs (the smart attacker model); under this new, more realistic threat model, the security features of the previous schemes proposed in the literature drastically decrease; (2) we provide a new pseudo-random key pre-deployment strategy that assures: (a) a key discovery phase that requires no communications; (b) high resilience against the smart attacker model.

We provide both analytical evaluations and extensive simulations of the proposed scheme. The results indicate that our pseudo-random key pre-deployment proposal achieves a provably efficient assignment of keys to sensors, an energy preserving key discovery phase, and is resilient against the smart attacker model.

1 Introduction

A Wireless Sensors Network (WSN) is a collection of sensors whose size can range from a few hundred to a few hundred thousands and possibly more sensors. The sensors do not rely on any pre-deployed network architecture, they thus communicate via an ad-hoc wireless network. Distributed in irregular patterns across remote and often hostile environments, sensors should autonomously aggregate into collaborative, peer-to-peer networks. Sensor networks must be robust and survivable despite individual sensor failures and intermittent connectivity (due, for instance, to a noisy channel or a shadow zone). WSNs are often infrastructure-less, and the power supply of each individual sensor is provided by a battery, whose consumption for both communication and computation activities must be optimized.

A WSN can be deployed in both military and civil scenarios [16, 2]. For instance, it could be used: (1) to provide a relay network for tactical communication in a battlefield; (2) to collect data from a field in order to reveal the presence of a toxic gas; (3) to facilitate rescue operations in wide open hostile areas; (4) to fulfill perimeter surveillance duties; (5) to operate for commercial purpose in severe environmental constrained scenarios (for instance, to measure the concentration of metals such as nodules of manganese on the ocean bed).

The key establishment phase, that is establishing secure pairwise communications, can be useful for many applications. In particular, it is a pre-requisite for the implementation of secure routing, and can be useful for the establishment of secure group communications. The key establishment problem has been widely studied in general network environments. There are three types of general key establishment schemes: trusted-server scheme, self-enforcing scheme, and key pre-deployment (a.k.a. key pre-distribution) scheme. The trusted-server scheme depends on a trusted server for key agreement between nodes, e.g., Kerberos [19]. This type of scheme is not suitable for sensor networks because of the lack of trusted infrastructure. The self-enforcing scheme relies on asymmetric cryptography, such as the use of public key certificates for home banking transactions. However, limited computation and energy resources of sensors often make it undesirable to use public
key algorithms, as pointed out in [20, 2]. As for key pre-
distribution scheme, the keys are distributed to all sensors
prior to the sensor deployment on the ground. If we knew
which sensors will be in the same neighborhood before the
sensor deployment, keys could be assigned to sensors a pri-
ori. However, such a priori knowledge often does not exist.

There exist a number of key pre-distribution schemes
which do not rely on a priori sensor deployment knowl-
edge [8]. A naïve solution is to let all the sensors carry a
master secret key. Any pair of sensors can use this global
master secret key to achieve key agreement and obtain a
new pairwise key. This scheme does not exhibit desirable
network resilience: if one sensor is compromised, the se-
curity of the entire sensor network is compromised. Some
existing studies suggest to store the master key in a sensor
tamper-resistant memory to reduce the risk [10, 5]. How-
ever a few drawbacks exist: (1) an increase in both the
per-sensor cost and energy consumption to manage tam-
per resistant modules; (2) tamper-resistant hardware might
not always be actually safe [4]. A second, simple key pre-
distribution scheme for implementing secure pairwise com-
munication is to assign a set of secret keys to each sensor,
each key of the set being shared with only one other sen-
ator. This solution requires each member to store \( n - 1 \) keys,
where \( n \) is the size of the group, with \( \frac{n(n-1)}{2} \) different keys in
the group. Here, the resilience is optimal because a compro-

mised sensor does not affect the security of other sensors;
however, this scheme is impractical for sensors as \( n \) can be
very large. Moreover, adding new sensors to a pre-existing
sensor network is not easy. To overcome all the above lim-
itations, pseudo-random key pre-deployment schemes have
been proposed [14, 9, 13, 17, 11]. Details on these schemes
in Section 2.

This paper introduces the smart attacker model that dra-
 matically decreases the level of confidentiality provided
by all pseudo-random key pre-deployment schemes pro-
posed in the literature. Then, we propose a novel key pre-
deployment scheme that: (1) is resilient to the smart at-
tacker; (2) requires minimal energy consumption in the key
discovery phase. Indeed, our scheme supports a key set up
that does not require any message exchange, only a lim-
ited number of hash function invocations. Moreover, the
proposed scheme can be applied to improve any solution
to establish pairwise keys based on random and pseudo-
random key pre-deployment, such as [14, 9, 13, 17, 11].

In Section 2, we review the current contributions in the
field. In Section 3 we introduce the smart attacker, a novel
threat model that undermines the resilience of all known
schemes. In Section 4 we detail all the key discovery phases
proposed in the literature and show their weaknesses when
facing the smart attacker model. In Section 5, we intro-
duce our novel key pre-deployment scheme and prove its
optimality in memory usage. Then, we focus on the ex-
perimental evaluation of our proposed scheme, in terms of
computational efficiency and resilience. Finally, Section 6
presents some concluding remarks.

2 Preliminaries and Related Work

We assume a WSN composed of \( n \) sensors. A sensor
is denoted by \( s \). To address a specific sensor, we write \( s_a \),
where \( a \) is the sensor’s id. Each sensor can hold \( K \) sym-
metric encryption keys. The center assigns keys to sensors
following a random (pseudo-random) key pre-deployment
strategy:

1. a pool of \( P \) random keys \( \{v_{p1}, \ldots, v_{pP}\} \) is generated;

2. for each sensor \( s_a \) in the WSN, a set \( V_a = \{v_{a1}, \ldots, v_{aK}\} \)
(called key-ring) of \( K \) distinct keys is pseudo-randomly
drawn from the pool and assigned to sensor \( s_a \).

When two sensors wants to communicate securely, they first
find out via a key-discovery phase which keys of the pool
they share, then they compute a channel key from a subset
of the shared keys. The channel key is used to secure the
channel by using a symmetric key encryption algorithm. We
denote with \( E_k(m) \) the encryption with key \( k \) of message \( m \)
and with \( E_k^{-1}(m) \) the decryption of message \( m \) with key \( k \).

In Eschenauer and Gligor [14], the idea of probabilistic
key sharing for WSN is firstly introduced. The authors pro-
vide a centralized algorithm for re-keying in a distributed
WSN. Before deployment, each sensor node receives a ran-
donm subset of keys from a large key pool; to agree on a key
for communication, two nodes find one common key within
their subsets and use that key as their shared secret key.

In [9], three mechanisms in the framework of random
key predistribution to address the bootstrapping problem are
proposed. First, the \( q \)-composite random key predistribution
scheme achieves probabilistic security under small scale at-
tack while trading off increased vulnerability in the face of
a large scale physical attack on network sensors. Second,
the multi-path key reinforcement scheme substantially in-
creases the security of key setup such that an attacker has
to compromise many more sensors, with respect to the \( q-
composite \) scheme, to achieve a high probability of com-
promising any given communication. Finally, the random-
pairwise keys scheme assures that, even when part of the
sensors have been compromised, the rest of the network re-
mains fully secure. These features have been achieved with
the following drawbacks: (1) the \( q \)-composite scheme is en-
ergy demanding, because of the key discovery phase that re-
quires a number of messages proportional to the number of
keys assigned to each sensor; (2) the multi-path reinforce-
ment scheme requires the availability of disjoint paths be-
tween the sender and the receiver. Disjoint multi-path find-
ing is an NP-hard problem, and moreover requires knowl-
edge of the topology; and (3) the sensor-to-sensor mutual authentication scheme does not scale.

In [21], the authors propose a protocol based on the key predistributed technique in [14]. The main idea used in this protocol is that the set of keys assigned to each sensor is computed using a pseudo-random generator with the id of the sensor as the seed. In this way it is possible for any sensor to know the set of keys stored by any other sensor. This feature supports a key discovery phase that involves only local computations to the sensor and no message exchange. The protocol admits two operations: group re-keying and threshold based sensor revocation. The group re-keying scheme, as well as the threshold based sensor revocation, are based on probabilistic key sharing. Later in this paper we show that this protocol is very weak against the novel smart attacker model.

In [13], the authors use a scheme proposed by Blom [6] that allows any pair of nodes in a network to be able to find a pairwise secret key. The salient feature of Blom’s scheme is that, as long as no more than $\lambda$ nodes are compromised, the network is perfectly secure (this is called the $\lambda$-secure property). A $\lambda$-secure data structure built in this way is called a key space. The authors create a set $\mathcal{W}$ composed of $\omega$ key spaces, and assign to each sensor up to $\tau$ spaces, randomly chosen in $\mathcal{W}$. Two nodes can find a common secret key if they have both picked a common key space. This phase, called by the authors key agreement phase, is carried out by making each node disclose some information about the key spaces it has been assigned, thus either disclosing the indexes of the spaces each node carries, or using the challenge response technique in [9]. In the first case, the scheme shows the same weaknesses of [21] against the smart attacker, while, in the second case, the scheme is more energy consuming since it requires at least $\tau$ messages for each key set up.

In [17], the authors develop a general framework for polynomial pool-based pairwise key predistribution in sensor networks based on basic polynomial-based key predistribution [7]. In principle, this work is very similar to [13], where Blundo et al.’s polynomial scheme is used in the place of Blom’s scheme. Finding the polynomial shared between two sensors is a crucial phase of the protocol. Again, the choice is between the weaker disclosing indexes techniques, and the energy demanding challenge response technique.

### 3 Security Model

Devising a security model requires to define the security **target** and a **threat model**. The security target identifies: the properties we want to protect and the assumptions about the working hypothesis. The threat model formulates hypothesis about the attacker capabilities and its possible behavior.

#### 3.1 Security Objectives

In this paper we focus on protecting confidentiality of the information. Our working hypothesis are the following:

- The cryptographic primitives employed are computationally secure [1, 18];
- the algorithms, protocols and mechanisms employed to secure the WSN are publicly known. Only keys in the sensors’ key-rings and in the pool are initially secret [3].

Since confidentiality is achieved encrypting communications with symmetric key encryption algorithms, confidentiality of the message translates into confidentiality of the key.

#### 3.2 The Threat Model

A common assumption is that the attacker is compliant with the Dolev-Yho model [12]. This means that the attacker can perform the following actions: (1) intercept and learn any message; (2) introduce forged messages into the system using all the available information. However, a distinguishing feature of WSNs is that sensors may be unattended. Hence, an additional and more powerful action the attacker can perform is to capture a sensor and acquire all the information it stores. To cope with this threat it is possible to assume sensors are tamper-proof, like in [5, 10]. However, a very large number of sensors can be built only if sensors are low-cost devices, and this makes it difficult for manufacturers to make them tamper-resistant. Therefore, in the rest of this paper we assume that sensors do not have tamper proof components and that can be captured. Moreover, we assume the attacker’s goal is to violate the confidentiality of the channel established between two arbitrarily chosen sensors $s_a$ and $s_b$. In order to achieve his goal, the attacker can compromise any subset of the sensors in the WSN but $s_a$ and $s_b$. To formalize the attacker in a random pre-deployment scheme we introduce the following definition.

**Definition 1.** Assume the attacker’s goal is to collect a subset $T$ of the keys in the pool. The attacker has already compromised a number of sensors, and has collected all their keys in a set $W$. The key information gain $G(s)$ of sensor $s$ is a random variable equal to the number of keys in the key-ring of $s$ which are in $T$ and are not in $W$.

Since we assume the attacker’s goal is to compromise the channel between sensors $s_a$ and $s_b$, subset $T$ in the above definition contains all the keys which are in the key-ring of both $s_a$ and $s_b$.

We define two attackers. Both of them tamper sensors sequentially in order to collect all the keys in $T$. 


1. The **oblivious attacker**: at each step of the attack sequence, the next sensor to tamper is chosen at random between those yet to be compromised.

2. the **smart attacker**: at each step of the attack sequence, the next sensor to tamper is sensor \( s \), where \( s \) maximizes \( E[G(s) | I(s)] \), the expectation of the key information gain \( G(s) \) given the information \( I(s) \) the attacker knows on sensor \( s \) key-ring.

Intuitively, the oblivious attacker does not take any advantage from any information leaked by the sensors. Conversely, the smart attacker greedily uses this information to choose the best sensor to corrupt in order to maximize the number of useful keys it is expected to collect. Note that information \( I(s) \) depends on the protocols used by sensor \( s \) and, in particular, on the key discovery phase. All previous works in the literature we know of assume the oblivious attacker.

### 4 Key Management

#### 4.1 The Key Discovery Phase

All the key deployment schemes reviewed in Section 2 require a key discovery phase. In particular, three different schemes have been proposed so far:

1. **key index notification**;
2. **challenge response**;
3. **pseudo random key index transformation**.

The computational and communication complexity of the above key discovery procedures is summarized in Table 1.

**Key index notification** [14] requires sensor \( s_a \) to send sensor \( s_b \) the indexes of the keys in its key-ring. Sensor \( s_b \), in turn, notifies sensor \( s_a \) of the indexes of the keys to be used to secure the channel. The communication complexity for sensor \( s_a \) is equal to \( K \) messages, where \( K \) is the size of the key-ring. The computational complexity for sensor \( s_b \) is also equal to \( K \), since we can assume that both sensor \( s_a \) and sensor \( s_b \) store their keys sorted according to their indexes.

**Challenge response** also introduced in [14] and later used in [9, 13, 17], does not leak any information about the indexes of the keys in the key-ring, at the price of an higher energy cost. Indeed, sensor \( s_a \) is supposed to broadcast messages \( \alpha, E_{s_a \text{-key}[i]}(\alpha), i = 1, \ldots, k \), where \( \alpha \) is a challenge and \( s_{a \text{-key}[i]} \) is the \( i^{th} \) key assigned to sensor \( s_a \) from the pool. The decryption of \( E_{s_a \text{-key}[i]}(\alpha) \) with the proper key by sensor \( s_a \) would reveal the challenge \( \alpha \) and the information that \( s_b \) shares that particular key with sensor \( s_a \). This key discovery procedure requires \( O(k^2) \) decryptions on the receiver side and \( k \) encryptions on the sender side. Moreover, at least \( K \) messages have to be sent and received.

As observed in [21], **pseudo-random key transformation** allows a more efficient key discovery procedure than random schemes like key index notification and challenge response. Given a sensor \( s_a \), the idea is to generate the indexes of the keys that will be statically assigned to \( s_a \) using a pseudo-random generator initialized with a public seed dependent on \( s_a \). Once the seed is known, the \( k \) indexes of the keys assigned to \( s_a \) can be computed by anyone. Note that the key discovery procedure supported by pseudo-random key transformation reveals only the indexes of the keys given to sensor \( s_a \), and does not leak any information on the keys themselves. This key discovery procedure requires no message exchange, at most \( k \) applications of the pseudo-random generator, and \( k \) look-ups in the local memory each consisting of \( \log k \) operations.

Both key index notification and pseudo-random key transformation have been criticized for revealing the key indexes of the keys in each sensor’s key-ring to the attacker [14, 9]. It is argued, according to intuition, that knowing such indexes allows the attacker to pick and tamper exactly those sensors whose key-rings are mostly useful to compromise a given target channel. However, neither mathematical nor experimental justification have been given to support this intuition. Indeed, all previous works in the area assume the oblivious attacker, which is not able to take any advantage from the information leaked by key index notification and pseudo-random key transformation; the smart attacker we introduce, which can indeed exploit additional information revealed by the key discovery phase of both key index notification and pseudo-random key transformation, is also stronger against challenge response. This fact has never been noted in the literature.

Whichever key pre-deployment and key discovery protocol is used, the strategy of the oblivious attacker does not change. Basically, it is a random attacker. Conversely, the strategy of the smart attacker does depend on the protocol used. First of all, note that both key index notification and pseudo-random key index transformation reveal the indexes of the keys in the key-ring during the key discovery phase. Since all sensors, if willing to communicate securely, have to perform a key discovery phase, we can assume that the attacker knows the indexes for all sensors \( s \) in the network. Therefore, \( G(s) | I(s) \) is equal to \( |T \setminus W| \) with probability 1, \( E[G(s) | I(s)] = |T \setminus W| \), and the smart attacker strategy is to compromise the sensor that gives the largest number of new keys in \( T \).

Consider now challenge response. Sensor \( s \), during one of its key discovery phases, leaks enough information to let the attacker know exactly which keys in its key-ring are in \( W \) and just how many outside \( W \). Indeed, anyone who has a particular key can discover whether sensor \( s \) has that key.
Table 1. Complexity of the key discovery phase schemes.

<table>
<thead>
<tr>
<th>Model</th>
<th>Sender</th>
<th>Receiver</th>
<th>Key-Ring Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Messages</td>
<td>Computations</td>
<td>Messages</td>
</tr>
<tr>
<td>pseudo-random index [21]</td>
<td>0</td>
<td>K log K + K hash</td>
<td>0</td>
</tr>
<tr>
<td>PRK [this paper]</td>
<td>0</td>
<td>K hash</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 1. Experimental results on index notification and pseudo-random key index transformation: Number of sensors to corrupt in order to compromise an arbitrary channel.

Figure 2. Experimental results on challenge response: Number of sensors to corrupt in order to compromise an arbitrary channel.

Take two sensors $s_a$ and $s_b$. Intuitively, if $|(V_a \setminus W) \cap T| > |(V_b \setminus W) \cap T|$, then the expected number of “useful” keys for the attacker is larger in the key-ring of $a$, compared with the key-ring of $b$. That is, $E[G(s_a) | I(s_a)] > E[G(s_b) | I(s_b)]$. The smart attacker strategy is now clear: at each step of the attack sequence, compromise the sensor that stores the largest number of keys which are not in $W$.

4.2 Experiments on the Oblivious and the Smart Attacker

In our experiments, we select two arbitrary sensors $s_a$ and $s_b$ and assume that the attacker’s goal is to compromise the channel between these two sensors. We also assume a pool size of 1,000 keys, and a WSN of 1,000 nodes.

First, we consider index notification and pseudo-random key transformation under the attack of both the oblivious and smart attacker. It is not necessary to evaluate these two schemes separately, in fact, it is easy to realize that they behave in the same manner from a security point of view. Our results, shown in Figure 1, confirm intuition: these schemes are very weak against the smart attacker. Take, as an example, a key-ring of 80 keys. While the oblivious attacker has to capture around 30 sensors in order to compromise the channel, the smart attacker can select as many as 3 sensors to get all the keys it needs to compromise the same channel. As the key-ring size gets larger, the difference between the two attackers gets somewhat smaller. This is not surprising since, at the extreme, when $K = P$ all sensors has the whole pool in their key-ring, and one sensor is enough to corrupt all channels for both attackers.

Second, consider challenge response. The experimental results are shown in Figure 2. While the oblivious attacker behaves just like in the previous experiment (it does not take any advantage from knowing the key indexes), the smart attacker shows that challenge response is stronger indeed than the other two schemes. This is the experimental evidence of the intuitive idea that hiding the key indexes can yield a much more secure channel. However, and this is not remarked upon in the literature, the smart attacker turns out to be considerably stronger against challenge response, too. Take, again, a key-ring size of 80 keys. While the oblivious attacker needs to compromise 30 sensors, only 20 sensors are enough for the smart attacker.

The above results shows that the commonly assumed
oblivious attacker is not always suited for evaluating the security of random pre-deployment schemes, and that the more realistic and powerful smart attacker can be used to better understand and evaluate the security properties of these protocols.

5 The Pre-Deployment Scheme

No key discovery protocol in the literature is completely satisfactory (see Table 1). Pseudo-random index transformation is more efficient than challenge response, which, in turn, is both intuitively and experimentally more secure. Our goal is to define a key pre-deployment scheme that supports an efficient and secure key discovery phase, as efficient as index notification (no message exchange) and as secure as challenge response.

Let’s start from one of the basic tools we need.

Definition 2 ([115]). A pseudo-random function is an efficient (deterministic) algorithm which given an h-bit seed, y, and an h-bit argument, x, returns an h-bit string, denoted by \( f_y(x) \), so that it is infeasible to distinguish the responses of \( f_y \), for a uniformly chosen y, from the responses of a truly random function.

We assume to have access to a pseudo-random function \( f_y \), where y is uniformly chosen. As a consequence of the above definition, the output of \( f_y \) can be dealt with as being random for all our purposes.

The pre-deployment scheme we propose (PRK) works as follows:

1. Consider a new sensor \( s_{a} \);
2. for all keys \( v'_{g} \) of the pool, compute \( z = f_y(a \parallel v'_{g}) \);
3. if and only if \( z \equiv 0 \mod (P/K) \), then put \( v'_{g} \) in the key ring \( V_{a} \) of sensor \( s_{a} \).

In the above scheme, we implicitly assume that \( h \), the size of the input of the pseudo-random function, is big enough for argument \( a \parallel v'_{g} \). We also assume, in order to avoid tedious details, that \( P/K \) divides \( 2^h \).

PRK supports a very efficient key discovery procedure. Consider a sensor \( s_{b} \), willing to know which keys it shares with sensor \( s_{a} \). For all keys \( v'_{g} \in V_{b} \), sensor \( s_{b} \) computes \( z = f_y(a \parallel v'_{g}) \), then sensor \( s_{b} \) knows that sensor \( s_{a} \) also has key \( v'_{g} \) if and only if \( z \equiv 0 \mod (P/K) \). Complexity is comparable to pseudo-random index transformation: no message exchange and \( K \) applications of the pseudo-random function. Moreover, only who already knows key \( v'_{g} \) can know whether sensor \( s_{a} \) has that key or not by computing \( z = f_y(a \parallel v'_{g}) \) and checking out if \( z \equiv 0 \mod (P/K) \). All other entities gets no information from \( z \). This is exactly the same information revealed by challenge response.

In order to claim that we achieved the goal of defining a both secure and efficient key discovery procedure, we still have to prove some important properties of our key pre-deployment scheme. Each key \( v'_{g} \) of the pool is assigned to sensor \( s_{a} \) with the following probability:

\[
\Pr[v'_{g} \in V_{a}] = \Pr[f_y(a \parallel v'_{g}) \equiv 0 \mod (P/K)] = K/P.
\]

Therefore, the average number of keys assigned to sensor \( s_{a} \) is

\[
E[|V_{a}|] = P \cdot \Pr[v'_{g} \in A] = K.
\]

However, two constraints have to be taken in account. Keying size \( |V_{a}| \) should not be larger than \( K \), the sensor’s memory size. This is the hardware constraint. Moreover, \( |V_{a}| \) should not be smaller than a fixed number of keys \( M \), which is chosen in such a way to guarantee that sensor \( s_{a} \) carries enough keys to secure its channels as required. This is the security constraint. Of course, it is perfectly possible that the above procedure fills key-ring \( V_{a} \) in such a way to violate either the hardware constraint or the security constraint.

In the first case, when \( K < |V_{a}| \), the key-ring does not fit the sensor memory. Note that it is not possible to simply discard some of the keys in \( V_{a} \), since sensor \( s_{b} \) would believe to share keys that are not actually shared with sensor \( s_{a} \). Therefore, the whole sensor \( s_{b} \) has to be discarded since it does not meet the hardware constraint (of course, the only thing to discard is just id \( a \), not the actual hardware). The same occurs when \( |V_{a}| < M \), sensor \( s_{a} \) does not meet the security requirements and, again, has to be discarded. Now, our problem is to guarantee that a new valid sensor is generated after discarding a small number of invalid sensors and using “almost all” the memory of the sensor. To guarantee the latter requirement, we ask \( M \) to be equal to \( K - o(K) \), thus saying that only a negligible number of key slots are allowed to be empty in each sensor’s key-ring. An optimal solution to this problem is presented in the following theorem.

Theorem 1. A sensor can be generated in time \( O(P) \) such that its key ring is composed of at least \( K - o(K) \) keys and at most \( K \) keys, with probability \( 1 - o(1) \).

Proof. Consider the procedure of generating a sensor \( s_{a} \) according to PRK. The computational cost is \( O(P) \), since it involves \( P \) invocations of pseudo-random function \( f_y \), where \( y \) is randomly and uniformly fixed. Define \( P \) random variables \( X_1, \ldots, X_P \) in the following way:

\[
X_i = \begin{cases} 
1 & \text{if pool key } v'_{g} \text{ is assigned to sensor } s_{a}, \\
0 & \text{otherwise}.
\end{cases}
\]

Random variables \( X_i \) are mutually independent. The sum of these variables \( X = \sum X_i \) tells how many keys are assigned to sensor \( s_{a} \). Clearly, \( \Pr[X_i = 1] = K/P \) and \( E[X] = K \). Our
claim is that, asymptotically, $X$ is close to $K$ with high probability.

By the Chernoff bound, the probability that the expected value of $X$ is at most a factor $\delta > 0$ far from $K$ is as follows:

$$\Pr[(1 - \delta)K \leq X \leq (1 + \delta)K] \geq 1 - 2e^{-\frac{K\delta^2}{3}}.$$ 

Take $\delta = o(1) = o(1/\sqrt{K})$, e. g. $\delta = 1/\log K$. Since

$$\lim_{K \to \infty} K\delta^2 = \infty,$$

then

$$\Pr[(1 - \delta)K \leq X \leq (1 + \delta)K] = 1 - o(1),$$

which is the claim.

5.1 Experiments on PRK

PRK, our pre-deployment strategy, assures an asymptotically optimal allocation of keys in the key ring of the sensors. In this section, we show that PRK performs very well also from a practical point of view. In particular, we want to evaluate the efficiency of the proposed scheme in terms of the amount of computation performed to generate a key pre-deployment for a WSN of a specific size.

We use SHA-1 to implement a pseudo-random function. The cost of generating a new sensor is equal to $P$ (10,000 in our simulations) invocations of SHA-1 on a small, constant size input. Our experiments shows how many sensors have to be generated to get 10,000 valid sensors. We consider different key-ring sizes, from 60 keys to 300 keys in steps of 10. For each size, we ask that at least 90%, 95%, and 99% of the key-ring must be full of keys in order to get a valid sensor. We performed extensive simulation, reported in Figure 3.

The experiments support the following considerations. PRK is very efficient also for WSNs and key-rings of practical sizes. As an example, only four sensor ids have to be discarded until a valid sensor id is found, when the security constraint is set to 95% of the key-ring size 60. By increasing the key-ring size to 300, this result improves, only three sensor ids are discarded until a good one is found. Note that the graph with security constraint of 99% is decreasing as the key-ring size increases as all other graphs in the figure: its saw-teeth look is only due toarithmetical rounding (1% of 190 is just 1). This example, together with the behavior of all graphs in Figure 3, supports the asymptotic result of Theorem 1 and confirms that the pseudo-random key pre-deployment scheme proposed is both computationally efficient and scalable.

Finally, consider the security properties of PRK. We perform the same experiment as we did with challenge response in Figure 2. The security constraint of PRK is set...
to 95%. By comparing Figure 4 and Figure 2, it is possible to see that there is no sensible difference between the security properties of the two schemes. Even without asking an extremely high security constraint, the security of PRK is practically indistinguishable from the security of challenge response.

6 Concluding Remarks

In this paper we introduce the smart attacker model that dramatically decreases the level of channel confidentiality provided by the key pre-deployment schemes so far proposed in the literature. We cope with this new threat by developing a pseudo-random key deployment scheme that is both energy saving and secure against the smart attacker. Through analytical evaluation and extensive simulations, we show that the pseudo-random key pre-deployment scheme minimizes the computations and the communications required for the key discovery phase and is highly resilient against the smart attacker model. Moreover, the proposed scheme can be adopted by any of the schemes so far proposed in the literature and based on pseudo-random key pre-deployment, such as [14, 9, 13, 17, 11], to improve either their resilience against the smart attacker, or their energy consumption.

References