A new class of Hash-Chain based key pre-distribution schemes for WSN

Walid Bechkit, Yacine Challal, Abdelmadjid Bouabdallah

A New Class of Hash-Chain Based Key Pre-distribution Schemes for WSN

Walid BECHKIT[1], Yacine CHALLAL[1], Abdelmadjid BOUABDALLAH[1]

Université de Technologie de Compiègne, Laboratoire HeuDiaSyc, UMR CNRS 7253, Compiègne, France

Abstract

In the last decade, we witness a proliferation of potential application domains of wireless sensor networks (WSN). Therefore, a host of research works have been conducted by both academic and industrial communities. Nevertheless, given the sensitivity of the potential applications that are generally tightly related to the physical world and may be human beings, a large scale deployment of WSN depends on the dependability provided by these emerging networks. Particularly, security emerges as a challenging issue in WSN because of the resource limitations. Key management is one of the required building blocks of many security services, such as confidentiality, authentication, etc. Unfortunately, public key based solutions, which provide efficient key management services in conventional networks, are unsuitable for WSN because of resource limitations. Symmetric key establishment is then one of the most suitable paradigms for securing wireless sensor networks.

In this paper, we tackle the resiliency of symmetric key pre-distribution schemes against node capture. We propose a hash-based mechanism which enhances the resiliency of key pre-distribution for WSN. Applied to existing key pre-distribution schemes, our solution gives birth to enhanced schemes which are more resilient against node capture attacks. We analyze and compare our solution against the existing schemes, with respect to some important criteria such as: the network resiliency against node capture, secure connectivity coverage, storage requirement, communication overhead and computation complexity. We show through analytical analysis that our solution enhances the network resiliency without introducing any new storage or communication overheads. Moreover, we show that our solution introduces insignificant computational overhead.

Keywords: wireless sensor networks, security, key management, resilience

1. Introduction

A Wireless Sensor Network (WSN) is a wireless network which is composed of a set of tiny autonomous sensor nodes with sensing, computation, and wireless communication capabilities [1]. The purpose of such networks is to collect information issued from a controlled environment or a target object and then send it to a base station usually called the sink. Because of size factor and cost considerations, wireless sensor networks suffer from resource constraints including energy, memory, computation power and communication bandwidth and range. Nowadays, wireless sensor networks are increasingly used in numerous fields such as military, medical, industrial and environmental sectors; they are more and more involved in several sensitive applications which require sophisticated security services. Due to the resource limitation, existing security solutions for conventional networks could not be used in wireless sensor networks. So, the security issues became then one of the main challenges for the resource restricted environment of WSN which requires new specific solutions.

Key management is a corner stone service for many security services such as confidentiality and authentication which are required to secure communications in wireless sensor networks. The establishment of secure links between nodes is then one of the most challenging problems in WSN. The public key based solutions, which provide efficient key management services in conventional networks, are unsuitable for wireless sensor networks because of the energy, computation and storage limitations. Some public key schemes have been implemented on real sensors [2][3], however most researchers believe that these techniques are still too heavyweight over actual sensors' technology because they induce an important communication and computation overhead [4]. Symmetric key establishment is then one of the most suitable paradigms for securing exchanges in wireless sensor networks. Because of the lack of infrastructure in WSN, we have usually no trusted third party which can attribute pairwise secret keys to neighboring nodes, that is why key pre-distribution is the most suitable paradigm for wireless sensor networks.

Many symmetric key pre-distribution schemes for wireless sensor networks have been proposed in literature [5][6][7][8][9][10][11][12][13]. We classify these schemes into two categories: *deterministic schemes* which ensure a to-

Email addresses: wbechkit@hds.utc.fr (Walid BECHKIT), ychallal@hds.utc.fr (Yacine CHALLAL), bouabdal@hds.utc.fr (Abdelmadjid BOUABDALLAH)

tal secure connectivity coverage and *probabilistic schemes* where secure connectivity is not guaranteed and conditioned by the existence of shared keys. In order to evaluate the performances of key pre-distribution schemes, we consider five main metrics which are: network resiliency against node capture, secure connectivity coverage, communication overhead, computation complexity and storage overhead. In this paper, we consider in particular the resiliency of symmetric key pre-distribution schemes. Existing research works which addressed the network resiliency either introduce an important storage and communication overhead or degrade the secure connectivity coverage. In contrast to these solutions, our goal is to enhance the resiliency of WSN key management schemes without introducing any new storage or communication overheads while maintaining a high secure connectivity coverage.

In this work, we propose a hash-based mechanism which can be applied to existing pool based key pre-distribution schemes to enhance the network resiliency. To achieve this goal, we introduce a new method based on one way hash chain which conceals keys in such a way that the disclosure of some keys reveals only derived versions, which cannot be used to compromise other links in the network using backward keys. We call our solution: Hash-Chain class of key pre-distribution schemes and denote it by HC. Our class, applied to existing key pre-distribution schemes, gives birth to new schemes which are more resilient against node capture attacks. We carried out analytical analysis to compare the efficiency of our approach against basic schemes with respect to important performance criteria: the network resiliency against node capture, the secure connectivity coverage, the communication overhead, the computation complexity and the storage overhead. The obtained results show that our solution enhances the network resiliency and reduces the fraction of compromised links up to 40% compared to existing schemes while it guarantees the same secure connectivity coverage, storage and communication performance. Moreover, we show through analytical analysis and simulations that the induced computational overhead is insignificant and that the energy consumed by hash computations in our solution is negligible.

The contributions of our work are many folds and can be summarized in the following points:

- We review the state of the art of key management in wireless sensor network. Then, we propose a classification of symmetric key management schemes for WSN into two categories: *probabilistic* schemes and *deterministic* ones. We further refine the classification into sub-categories with respect to the underlying concepts and techniques used in key exchange and agreement.

- We introduce a mechanism to enhance the network resiliency of key pre-distribution schemes for WSN. The proposed solution is based on lightweight hash chains and improves the resiliency of existing pool-based key pre-distribution schemes.

- We analyze and compare our solution against the existing schemes, with respect to important criteria which are: the network resiliency against node capture, secure connectivity coverage, storage requirement, communication overhead and computation complexity. We show through analytical analysis that our solution enhances the network resiliency without introducing any new storage or communication overheads. Moreover, we show that our solution does not introduce significant computational overhead.

The remainder of this paper is organized as follows: section 2 presents related works on key management for wireless sensor networks. We define in section 3 the metrics used to evaluate our solution and to compare it to existing schemes. Section 4 gives a general idea of our class of pool based key pre-distribution schemes. In section 5, we present the resilient HC(q-composite) scheme issued from the application of our mechanism to the well known probabilistic q-composite scheme , we compare the performances of the two schemes and we discuss analytical results. In section 6, we apply our class to a deterministic scheme based on the combinatorial design which gives birth to a more resilient deterministic scheme; we also analyze and compare their performances. In section 7, we introduce a new smart attack against our mechanism and propose an enhanced version as countermeasures against this attack. Finally, section 8 ends up this paper with some conclusions.

2. Related works: key management schemes for WSN

Key management problem in wireless sensor networks have been extensively studied in the literature and several solutions have been proposed. Many classifications of symmetric key management schemes can be found in [14][15][16]. In this work, we mainly classify symmetric schemes into two categories: *probabilistic* schemes and *deterministic* ones. In *deterministic* schemes, each two neighboring nodes are able to establish a direct secure link which ensures a total secure connectivity coverage. In *probabilistic* schemes, the secure connectivity is not guaranteed because it is conditioned by the existence of shared keys between neighboring nodes.

2.1. Probabilistic schemes

In probabilistic key management schemes, each two neighboring nodes can establish a secure link with some probability. If two neighboring nodes cannot establish a secure link, they establish a secure path composed of successive secure links.

Eschenauer and Gligor proposed in [5] the basic Random Key Pre-distribution scheme denoted by RKP. In this

scheme, each node is pre-loaded with a key ring of m keys randomly selected from a large pool S of keys. After the deployment step, each node i exchanges with each of its neighbor j the list of key identifiers that it maintains. This allows node j to identify the keys that it shares with node i. If two neighbors share at least one key, they establish a secure link and compute their session secret key which is one of the common keys. Otherwise, nodes i and j do not have common keys. So, they should determine secure paths which are composed by successive secure links. The values of the key ring size m and the key pool size $|S|$ are chosen in such a way that the intersection of two key rings is not empty with a high probability p. This basic approach is CPU and energy efficient but it requires a large memory space to store the key ring. Moreover, if the network nodes are progressively corrupted, the attacker may discover a large part or the whole global key pool. Hence, a great number of links will be compromised. This is due to the fact that a given key may be used to secure different links if it is common between key rings of different pairs of nodes.

Chan et al. proposed in [6] a protocol called q-composite scheme that enhances the resilience of RKP. In this solution, two neighboring nodes can establish a secure link only if they share at least q keys. The pairwise session key is calculated as the hash of all shared keys concatenated to each other: $K_{i,j} = Hash(K_{s_1} \| K_{s_2} \| ... \| K_{s_{q'}})$ where $K_{s_1}, K_{s_2}, ...K_{s_{q'}}$ are the q' shared keys between the two nodes i and j ($q' \geq q$). This approach enhances the resilience against node capture attacks because the attacker needs more overlap keys to break a secure link. However, this approach degrades the network secure connectivity coverage because neighboring nodes must have at least q common keys to establish a secure link.

Chan et al. proposed in [6] another pairwise key predistribution scheme in which they aim to ensure a perfect resiliency and a given secure coverage probability p. Authors propose to use a distinct key to secure each link between each pair of nodes. They assign to each two neighboring nodes i and j a distinct key $k_{i,j}$. Prior to deployment, each node is pre-loaded with $p * N$ keys, where N is the network size (number of nodes in the network) and p is the desired secure coverage probability. Hence, the probability that the key $k_{i,j}$ belongs to the key set of the node i is p. Since we use distinct keys to secure each pair-wise link, the resiliency against node capture is perfect and any node that is captured reveals no information about links that are not directly connected to it. It is obvious that the main drawback of this scheme is the non scalability because the number of the stored keys depends linearly on the network size. In addition, this solution does not allow the node post-deployment because existing nodes do not have the new nodes' keys.

Du et al. proposed in [7] to enhance the secure connectivity and the resiliency of the basic random key predistribution schemes. Their solution requires deployment knowledge and uses several key pools instead of one. Nodes are organized in regional groups to which is assigned different key pools and each node selects its m keys from the corresponding key pool. The key pools are constructed in such a way that neighboring key pools share more keys while key pools far away from each other share fewer keys or no keys at all. This approach allows to enhance the secure connectivity coverage because the key pools become smaller. Moreover, the resiliency of Du et al. solution is improved since if some nodes of a given region are captured, the attacker could discover only a part of the corresponding group key pool. However, the application of this scheme is restrictive since the deployment knowledge of the wireless sensor network is not always possible.

In [8], Castelluccia and Spognardi adapted the basic scheme described above to the multi-stage wireless sensor networks where new nodes are periodically deployed to ensure the network connectivity. With a one-stage deployment, if the network is continuously attacked, the attacker could discover an important part of the global key pool and new established links will be immediately compromised. Authors in [8] proposed the use of keys with limited lifetimes and which are updated periodically. The proposed solution divides the lifetime of a node into k generations. Initially, each node has two key rings: a forward key ring (FKR) and a backward one (BKR). The forward key rings are updated at each generation using a secure hash function while the backward ones are updated using a hash-based Lamport chain, where they start by generating the keys of the last expected generation. If a node is deployed at generation i and will last at most until generation j, it will be configured with the FKR of generation i and the BKP of the generation j. The two key subsets are updated at each generation thanks to the two hash functions. Authors in [8] showed that by using limited lifetime keys, the network self-heals and recovers its initial state when attacks stop. In the other hand, the ratio of compromised links remains constant when the network is constantly attacked in contrast to the basic schemes.

In [9], Liu et Ning proposed a new pool based polynomial pre-distribution scheme for WSN based on previous work of Blundo et al. [17]. This approach can be considered as an extension of the basic RKP scheme where nodes are pre-loaded with polynomials instead of keys. Basic polynomial key management scheme proposed in [17] is based on symmetric bivariate λ-degree polynomials P(x,y) generated over a finite field F_q where q is prime. The main property of symmetric bivariate polynomial is that $P(x, y) = P(y, x)$. In this scheme, each node i is pre-loaded with the polynomial $P_i(y) = P(i, y)$. In order to establish a secure link between two neighbors i and j, each node evaluates its polynomial at the identifier of the other node. Hence the secret key between two nodes i and j is $K_{ij} = P_i(j) = P_j(i)$. The node identifiers are supposed to be unique. This approach was proved to be secure and λ-collusion resistant. In [9], Liu and Ning extended this approach to wireless sensor networks when they proposed a pool based polynomial pre-distribution scheme to establish

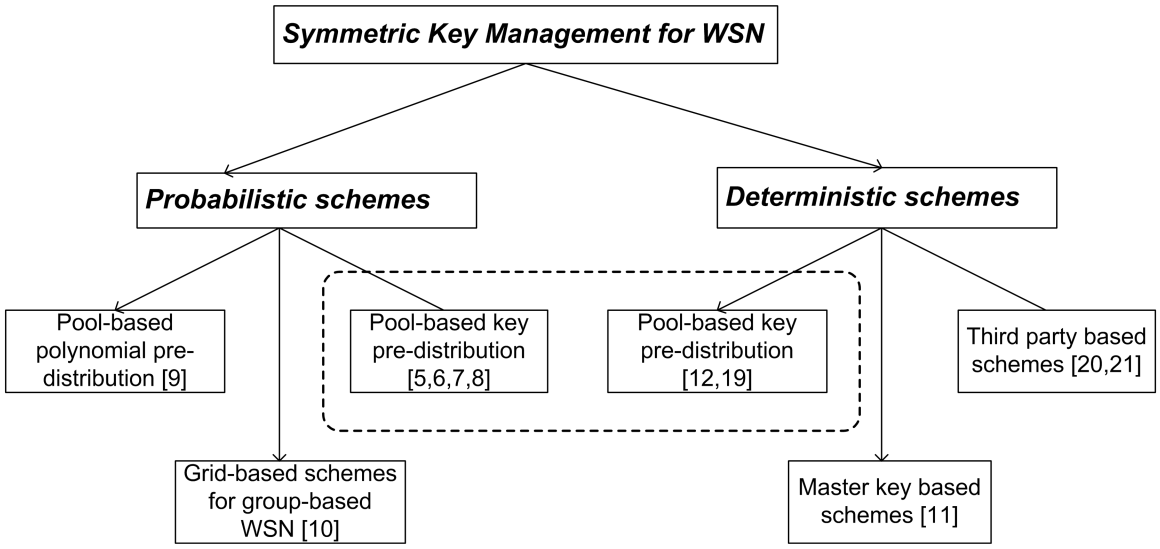


Figure 1: Classification of symmetric key management schemes for WSN

secure links. A global pool of symmetric bivariate polynomials is generated off-line and each node is pre-loaded prior to deployment with a subset of polynomials. If two neighboring nodes share a common polynomial, they establish a direct secure link as described above; else, they try to find a secure path as proposed in the RKP scheme. This approach allows to compute distinct secret session, so the resilience against node capture is enhanced. However, this solution requires more memory to store the symmetric bivariate polynomials and induces more computational overhead.

In [18], Blom proposed a symmetric key generation system in which each node i stores a column i and a row i of size $(\lambda + 1)$ of two matrices G and $(D * G)^T$ respectively where : $D_{(\lambda+1)\times(\lambda+1)}$ is a symmetric matrix, $G_{(\lambda+1)\times n}$ is a public matrix and $(D * G)^T$ is a secret matrix. The matrix of pairwise keys of a group of n nodes is then $K = (D * G)^T G$. In other terms, each pair of nodes i and j compute their session key as $K_{ij} = row_i * column_j = row_j * column_i$. This solution is proved to be λ-secure, which means that keys are secure if no more than λ nodes are compromised. The Blom's scheme cannot be applied to all the network nodes in a wireless sensor network because each node needs memory to store huge vectors. Yu and Guan [10] adapted the Blom's scheme to group-based wireless sensor networks where they assume deployment knowledge. Nodes are deployed into a grid and each group is assigned a distinct secret matrix. Using deployment knowledge, the potential number of neighboring nodes decreases which requires less memory. We believe that this scheme is only adapted for some restrictive applications of wireless sensor networks since deployment knowledge is not always possible.

2.2. Deterministic schemes

Deterministic schemes ensure that each node is able to establish a pair-wise key with all its neighbors. Many solutions were proposed to guarantee determinism.

LEAP [11] make use of a common transitory key which is preloaded into all nodes prior to deployment of the WSN. The transitory key is used to generate pairwise session keys and is cleared from the memory of nodes by the end of a short time interval after their deployment. LEAP is based on the assumption that a sensor node, after its deployment, is secure during a time T_{min} and cannot be compromised during this period of time. LEAP is vulnerable to node compromise. For instance, if the transitory initial key is discovered, the entire network could be compromised.

In [12], Çamtepe and Yener proposed a deterministic pool based key pre-distribution scheme. Instead to select randomly a subset of keys from the global key pool, they propose a new construction methodology based on combinatorial design theory. The main purpose is that each two subsets of keys share exactly one common key which ensures a total secure connectivity coverage. To achieve this goal, authors make use of particular combinatorial design which is the Symmetric Balanced Incomplete Block Design theory. This scheme is presented with more details in section 6.

In [19], authors propose to use the Çamtepe scheme for key management in grid group WSN. Authors divide the deployment area into square regions. In each region, they propose to use the symmetric Balanced Incomplete Block Design based key pre-distribution in order to ensure the intra-region secure communications. Inter-region communications are guaranteed thanks to special nodes having plentiful resources and called agents. Furthermore, authors propose to enhance the Çamtepe scheme in order to avoid key identifier exchanges. For this purpose, they index all nodes and keys and propose a mapping between

node indexes and key indexes.

Perrig et al. proposed in [20][21] a security suite for WSN having two blocks: i) $SNEP$ which provides data confidentiality, two-party data authentication, and data freshness and ii) $\mu TESLA$ which provides authenticated broadcast. SPINS propose to use the base station (Sink) as a trusted third party (TTP) in order to establish secure links between nodes. If a node i wants to establish a session secret key with node j, it sends a request message to node j. Upon receiving this message, the node j sends a message to the sink (TTP) which verifies the authentication, generates the secret session key and sends it to the nodes i and j. The use of the sink as a third trust party leads to an important communication overhead and degrades the network performances.

We focus in this paper on probabilistic and deterministic pool based key management schemes as illustrated in figure 1. In what follows, we propose a mechanism to enhance the resilience of these schemes by concealing keys through the use of an efficient hash chaining mechanism. Our approach can be applied to any pool based key predistribution scheme including [5], [6], [7], [8], [12] and all schemes derived from them. This allows to define a new resilient class of pool based key pre-distribution schemes. In this paper, we apply our class to two schemes: the well known probabilistic q-composite scheme proposed by Chan et al. [6] and the deterministic scheme based on Symmetric Balanced Incomplete Block Design theory proposed by Çamtepe and Yener [12]. We evaluate our hash chain class and compare it against the basic schemes with respect to important performance metrics that we define in the next section.

3. Evaluation Metrics

In this work, we consider five metrics to evaluate performances of wireless sensor network key management schemes:

Network resiliency against node capture : Due to the resource limitations in WSN, sensor nodes are usually not tamper resistant. If an adversary compromises a node, he can read all secret information from its memory. Such an attack can compromise not only adjacent links of compromised links but also external links that are independent of the compromised nodes. We define the resilience against node capture R_x as the fraction of uncompromised external links when x sensor nodes are compromised. We note that we consider in a first time the oblivious attacks where nodes are randomly captured. We discuss later in section 7, possible smart attacks as well as countermeasures.

Secure connectivity coverage : We define the secure connectivity coverage as the fraction of secured direct links among possible links in the network which is the probability that a given pair of neighboring nodes are able to establish a secure link. Note that we focus in our metric on the direct (one hop) secure link establishment; some solutions propose to establish a multi-hop secure path when the direct secure link establishment fails which we do not consider in this metric.

Computation complexity : Some key management schemes like [6], [9], [10] and the approach that we propose in this work require to perform additional computational operations such as hash functions or modulo multiplications. The additional processing may consume more energy and may induce a computational delay. We calculate in this metric the average number of additional operations required to establish secure links.

Communication overhead : The communication overhead measures the size of data exchanged between a pair of nodes in order to establish direct secure links. This metric has a significant influence on the energy consumption because the most of the consumed energy is due to communication.

Storage overhead : Because of their small size, sensor nodes are very constrained in term of memory resource. In what relates to key management schemes, we focus on the memory required to store the keys.

The proposed classification of key pre-distribution schemes into deterministic schemes and probabilistic ones is based on the secure connectivity coverage metric. Deterministic schemes are those which ensure a total secure connectivity coverage (the probability of establishing direct secure links is one). However, the probabilistic schemes are those which have not a total secure connectivity coverage.

4. Our Solution: HC(x) A resilient class of hash-chain based key pre-distribution schemes

As introduced before, we consider in this work the resiliency of symmetric key pre-distribution schemes against node capture. We propose a resilient class of key predistribution protocols that we denote by HC(x) for HashChain(x), where x is an existing pool based key predistribution protocol. This class of protocols enhances the resilience of existing schemes through a lightweight hash chaining technique that conceals the same keys pre-loaded in different sensor nodes. A preliminary work and few discussions on the network resiliency were presented in [22]. To conceal keys, we apply a one way hash function to the pre-distributed keys before deployment: a hash function h is applied to each node keys a number of times depending on the node identifier. As known, the main characteristic of hash functions is that knowing a value of the chain it is computationally infeasible to determine the backward values. So, with our approach, when an attacker corrupts one or more keys it can only discover a derived version (forward values).

Let us consider two neighboring nodes i and j. Node i applies i times the hash function h to each key of its

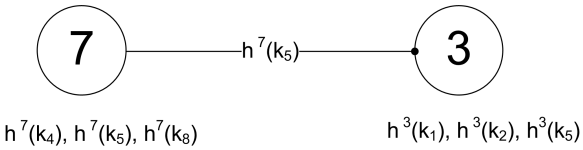


Figure 2: Illustration of the hash-chain based approach

key ring. Node j applies j times the hash function h to each key of its key ring. When nodes i and j discover a common key identifier id, the common key between i and j is then $h^{max(i,j)}(K_{id})$). Node $min(i,j)$ can calculate the common key by applying the hash function h, $|i-j|$ times to the key having the identifier id. Figure 2 illustrates these steps applied to two nodes whose identifiers are 3, and 7 respectively. We notice that key having the identifier 5 is common to both nodes 3 and 7. The common key is then $h^7(K_5)$ and node 3 can then calculate this common key by applying again the hash function h, $|3-7| = 4$ times to its key $h^3(K_5)$.

The application of this version of our class in large scale wireless sensor network may introduce an important hash computation overhead. Indeed, nodes may have high identifiers and should then apply the hash function a large number of times. In order to reduce the computation overhead, we introduce a parameter called L. Each node i applies the one way hash function h to their keys (i mod L) times instead of i times, the number of hash computation is then bounded by L. We discuss later how to choose the L parameter and its impact on the network resiliency and the hash computation overhead.
We show in what follows that we enhance the network resiliency against node capture where the computational overhead remains insignificant. Our approach can be applied to any pool based key pre-distribution scheme. To facilitate the explanation of the concepts behind our approach, we have chosen to apply our mechanism to two concrete schemes. A probabilistic scheme which is the q-composite scheme [6] and a deterministic one which is the Symmetric Balanced Incomplete Block Design scheme [12].

5. HC(q-composite): A highly resilient q-composite scheme

In this section, we develop a highly resilient key management solution by applying our solution to the q-composite scheme [6]. The proposed protocol which we denote by HC(q-composite) is more resilient against node capture as we demonstrate in the following analysis.

Before the deployment of the WSN, a large pool S of keys and their identifiers are generated off-line. Each node is preloaded with a key ring of m keys randomly selected from the key pool S. Before the deployment phase, we apply a hash function h, (i mod L) times to the pre-loaded keys of each node, where i is the node identifier and L is a parameter of our class, which allows to reduce the number of hash operation as explained before. So, each node i is preloaded with the set of keys KR^i such us:
$KR^i = \left\{h^{i \ mod \ L}(K_1), h^{i \ mod \ L}(K_2), ...h^{i \ mod \ L}(K_m)\right\}$
where $K_1, K_2, ...K_m$ are the randomly selected keys from S.

After the deployment phase, each two neighboring nodes can establish a secure link only if they share q or more common keys. The pairwise secret key between two neighboring nodes is computed as the hash of all their shared keys concatenated to each other.
Let us assume that the node i shares q' keys ($q' >= q$) with its neighbor j and that (i mod L) > (j mod L).
The node i computes its shared secret key with j as follows:

$K_{ij} = Hash(K^i_{s_1} \| K^i_{s_2} \| ... \| K^i_{s_{q'}})$

where $K^i_{s_1}, K^i_{s_2}, ..., K^i_{s_{q'}}$ are the common keys with the node j.

The node j computes its key as :

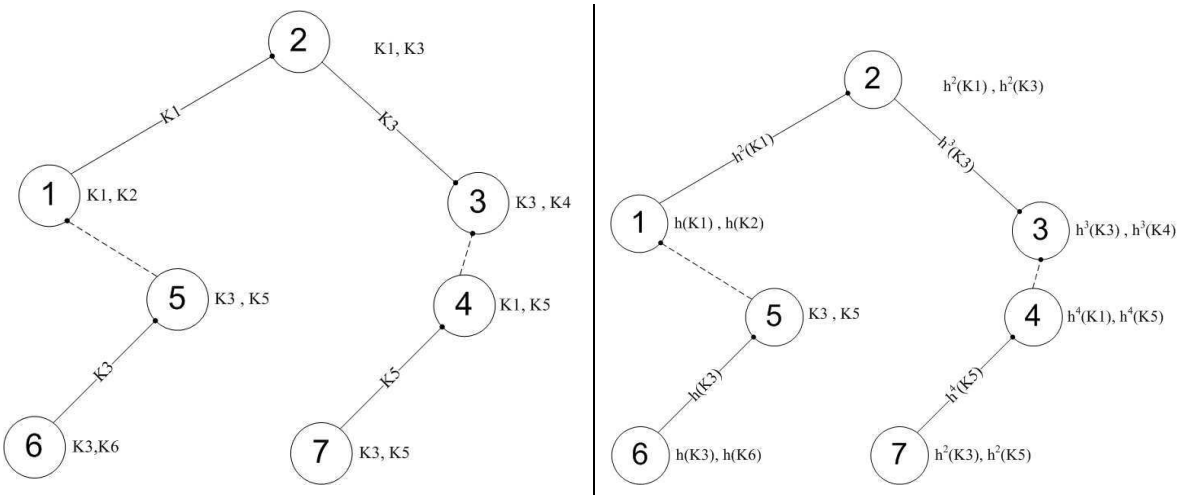
$$\begin{aligned} K_{ji} \quad = \quad & Hash(h^{(i \ mod \ L)-(j \ mod \ L)}(K^j_{s_1}) \| \\ & h^{(i \ mod \ L)-(j \ mod \ L)}(K^j_{s_2}) \| \\ & ... \| h^{(i \ mod \ L)-(j \ mod \ L)}(K^j_{s_{q'}})) \end{aligned}$$

where $K^j_{s_1}, K^j_{s_2}, ..., K^j_{s_{q'}}$ are the common keys with the node i. Indeed, node j which have the minimum identifier may apply the hash function (i mod L) − (j mod L) times to each shared key ($K^j_{s_p}$) in order to compute the key having the same identifier and pre-loaded in node i which is ($K^i_{s_p}$). The session key between i and j is then $K_{ij} = K_{ji}$.

5.1. Example:

To illustrate our idea, let us refer to the figure 3. To simplify the comprehension, we apply our approach to the 1-composite scheme (q-composite scheme with q=1) and we consider a network of seven nodes, a key pool containing six keys ($|S| = 6$) and each node is pre-loaded with $m = 2$ keys. In the basic 1-composite scheme (Figure 3(a)), the corruption of nodes 4 and 7 induces the disclosure of the keys K_1,K_3 and K_5, and then the compromise of the three external links (1,2),(2,3) and (5,6).

Using our protocol HC(1-composite) (Figure 3(b)), keys are hashed before the deployment phase (we assume in the example that $L = 5$). So the node 4 is pre-loaded with the keys $h^4(K_1)$ and $h^4(K_5)$ instead of K_1 and K_5, the node 7 is pre-loaded with the keys $h^2(K_3)$ and $h^2(K_5)$ instead of K_3 and K_5 and so on. If we consider the same scenario as above, the corruption of the two nodes 4 and 7 induces the disclosure of the derived keys $h^4(K_1)$, $h^4(K_5)$, $h^2(K_3)$ and $h^2(K_5)$. In this case, the link (2,3) can be compromised because the attacker can compute $h^3(K_3)$ knowing $h^2(K_3)$. However, the two other external links (1,2) and (5,6) cannot be compromised because it is infeasible to calculate $h^2(K_1)$ knowing $h^4(K_1)$ and $h(K_3)$ knowing $h^2(K_3)$.



(a) 1-composite

(b) HC(1-composite)

Figure 3: Example: 1-composite Vs HC(1-composite)

5.2. Analysis

In this section, we will evaluate the performance of our protocol HC(q-composite), we compare it to the basic q-composite scheme with respect to the metrics defined in section 3. We recall in table 1 the main symbols that we use in what follows:

Table 1: SUMMARY OF NOTATIONS

S	The global key pool
$\|S\|$	The size of the global key pool
m	The size of the node key ring
q	The minimum number of common keys required to establish a secure link
$p(k)$	The probability that two nodes share exactly k keys in their subset of keys
p	The probability that two nodes can establish a secure link
L	The parameter of the HC approach
n	The network size (number of nodes)

5.2.1. Network resiliency against node capture

In this section, we compare the network resiliency of the two schemes: q-composite and HC(q-composite). As stated before, we define the network resiliency R_x as the fraction of uncompromised links when x nodes are captured, we have then:
$R_x = 1 - P(LC|NC_x)$
where :

- LC is the event that a link is compromised
- NC_x is the event that x nodes are compromised

Let us compute R_x when we use the q-composite and the HC(q-composite) scheme and let us first compute $P(LC|NC_x)$ in the two cases. We recall that we consider only external links which are independent of the compromised nodes.

When x nodes are captured, all their keys are compromised. In the two schemes, q-composite and HC(q-composite), each external secure link is computed as the hash of k keys concatenated to each other where $q \leq k \leq m$. The probability of a link compromise when x nodes are captured is then given as follows :

$$P(LC|NC_x) = \sum_{k=q}^{m} P(LC_k|NC_x)P(LS_k) \qquad (1)$$

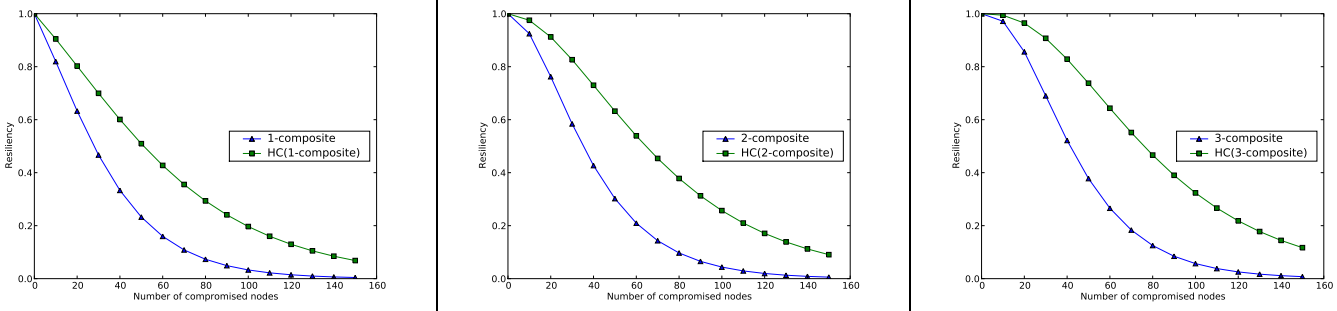
where:

- LC_k is the event that a link secured with k keys is compromised
- LS_k is the event that a secure link is secured with k keys

Proposition 1. *In the basic q-composite scheme, the probability of link compromise when x nodes are captured is :*

$$P(LC|NC_x) = \sum_{k=q}^{m} (1 - (1 - \frac{m}{|S|})^x)^k \frac{p(k)}{p}$$

Proof. In the basic q-composite scheme, the probability that a key is compromised when a node is captured is $c = \frac{m}{|S|}$ because each node is pre-loaded with m keys selected randomly from the global key pool S.
The fraction of uncompromised keys is then $1 - c$. When x nodes are compromised, the fraction of uncompromised



(a) q=1 (b) q=2 (c) q=3

Figure 4: Network resiliency depending on the number of compromised nodes (q-composite Vs HC(q-composite))

keys is $(1-c)^x$. So, the probability that a given key has been known is then $1-(1-c)^x$.
As a result, the probability that a given link is compromised when it is secured with k shared keys and when x nodes are compromised is:

$$P(LC_k|NC_x) = (1-(1-c)^x)^k \tag{2}$$

On the other hand, $p(k)$ which is the probability that two nodes share exactly k keys was given in [6] as :

$$p(k) = \frac{\binom{|S|}{k}\binom{|S|-k}{2(m-k)}\binom{2(m-k)}{m-k}}{\binom{|S|}{m}^2}$$

The probability that two given nodes can establish a secure link is then: $p = \sum_{k=q}^{m} p(k)$ and the probability that a secure link is secured exactly with k keys is:

$$P(LS_k) = \frac{p(k)}{p} \tag{3}$$

From equation (2) and (3) the equation (1) becomes:

$$P(LC|NC_x) = \sum_{k=q}^{m} (1-(1-\frac{m}{|S|})^x)^k \frac{p(k)}{p} \tag{4}$$

□

The resulting formula (4) is similar to the one calculated by Chan et al. in [6]. Let us now compute $P(LC|NC_x)$ of our HC(q-composite) scheme.

Lemma 1. *Using the hash chain class with $L \geq 1$, The probability that a given key is known when a node is randomly captured is:*

$$c_h = \frac{L+1}{2L}\frac{m}{|S|}$$

Proof. For any discovered key, it is initially hashed l times $(0 \leq l \leq L-1)$ with a probability $\frac{1}{L}$. When the initial key is hashed l times, the probability that a key having the same identifier can be discovered is $\frac{L-l}{L}$. Indeed, forward values can be computed when it is impossible to compute backward values.
Thus, we find that the probability to disclose a key having the same identifier with a compromised key is :

$$\sum_{l=0}^{L-1} \frac{1}{L} \times \frac{L-l}{L} = \frac{L+1}{2L}$$

Since each node has m keys, the probability that a key has been discovered when a node is compromised is $c_h = \frac{L+1}{2L}\frac{m}{|S|}$. □

Proposition 2. *In the HC(q-composite) scheme, the probability of link compromise when x nodes are randomly captured is :*

$$P(LC|NC_x) = \sum_{k=q}^{m} (1-(1-\frac{L+1}{2L}\frac{m}{|S|})^x)^k \frac{p(k)}{p}$$

Proof. Based on the lemma 1 and following the same steps of the proof of proposition 1, we find that using our HC(q-composite) scheme, the fraction of compromised links when x nodes are captured is :

$$P(LC|NC_x) = \sum_{k=q}^{m} (1-(1-\frac{L+1}{2L}\frac{m}{|S|})^x)^k \frac{p(k)}{p} \tag{5}$$

□

Propositions 1 and 2 show that the probability of link compromise is reduced when we apply our hash chain mechanism. Thus, our scheme is more resilient than the basic one. As we have defined resiliency against node capture as the fraction of uncompromised links when x nodes are compromised, we plot in the figure 4 the fraction of uncompromised links depending on the number of captured nodes with the two schemes (q-composite and HC(q-composite) when q=1,2 and 3. In this analysis, we fixed $|S| = 1000$, $m = 40$ and $L = 10$ (we discuss the choice of L's value

later). This figure shows clearly that the resilience against node capture attacks is better when using our approach. For instance, with $q = 3$, when the number of compromised nodes is between 50 and 100, our approach reduces the fraction of compromised links up to 40%. The figure shows also that higher is the q value, better is the improvement.

5.2.2. Secure connectivity coverage

Since we maintain the same common key discovery phase based on the key identifier exchange, the secure connectivity coverage of the network when we use hash chain q-composite scheme is the same as the basic q-composite scheme. Indeed, each two neighbors can establish a secure link only if they share at least q keys. The secure connectivity is then given as:

$$p_{connect} = p = \sum_{k=q}^{m} p(k)$$

We plot in figure 5 (a) the secure connectivity coverage depending on the key ring size m when q=1 ,2 and 3 and $|S| = 1000$. The figure shows that the secure connectivity coverage increases when the key ring size increases because the probability of sharing at least q keys increases. Moreover, the figure shows that higher is q worse is the secure connectivity coverage. For instance, with $q = 1$ and $m = 40$, we ensure a network secure connectivity coverage of 80%. To reach the same connectivity, we need about 55 keys per ring when $q = 2$ and about 65 keys when $q = 3$. Indeed, the q value represents the minimum required number of common keys to establish secure links.

In addition to the secure connectivity coverage, we plot also in figure 5 (b) the network resiliency when 50 nodes are compromised (R_{50}) depending on the key ring size m when $q = 2$, $|S| = 1000$ and $L = 10$. This figure shows that the network resiliency decreases when m increases. It shows clearly that our solution is more resilient against node capture attacks depending on the key ring size. Indeed, the network resiliency is improved up to 40% when $m \geq 60$.

5.2.3. Computation overhead

In the basic q-composite scheme, nodes apply a hash function one time on the common keys concatenated to each other in order to compute the secret key. With our solution, the node having the minimum hash degree should apply the hash function h a number of times before computing the secret key. In the example 3 (b) where $L = 5$, the node 7 for example should apply two times the hash function to compute $h^4(K_5)$ because it is initially pre-loaded with $h^2(K_5)$.
In order to evaluate this extra-computation overhead, we calculate the average number of times that a node applies the hash function per secure link in the initialization phase of our approach.

Lemma 2. *Using the hash chain approach with ($L \geq 1$), if two nodes can establish a secure link, then they have to apply on average $NH_{avg} = \frac{L^2-1}{6L}$ times the hash function on their common keys before computing the session secret key.*

PROOF. Let us assume that two nodes i and j can establish a secure link (which means that they share q' keys with $q' > q$).So, the node i is pre-loaded with keys hashed ($i \ mod \ L$) times where the node j is pre-loaded with keys hashed ($j \ mod \ L$) times. One of the nodes i or j would have to apply $|(i \ mod \ L) - (j \ mod \ L)|$ times the hash function on their q' common keys. Note that i and j are natural numbers chosen randomly, which means that ($i \ mod \ L$) and ($j \ mod \ L$) are uniformly distributed over Z_L. The average number of hash computation is then:

$$\begin{aligned}
NH_{avg} &= \frac{1}{2}\sum_{k=0}^{L-1} k \times P(|(i \ mod \ L) - (j \ mod \ L)| = k) \\
&= \frac{1}{2}(\sum_{k=1}^{L-1} k \times \frac{2(L-k)}{L^2}) \\
&= \frac{1}{L^2}(\sum_{k=1}^{L-1} k \times L - \sum_{k=1}^{L-1} k^2) \\
&= \frac{1}{L^2}(\frac{L^2(L-1)}{2} - \frac{L(L-1)(2L-1)}{6}) \\
&= \frac{L^2-1}{6L}
\end{aligned}$$

□

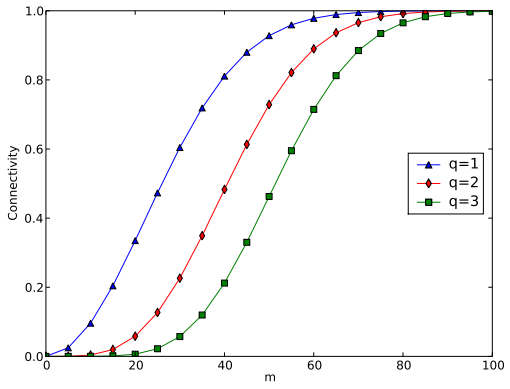
Proposition 3. *In the HC(q-composite) scheme, Given L and m, if two nodes can establish a secure link, then they have to apply on average $\frac{L^2-1}{6L}\sum_{k=q}^{m}(k \times \frac{p(k)}{p})$ hash functions.*

PROOF. Following the lemma 2, when using the HC(q-composite) scheme, each node applies on average $\frac{L^2-1}{6L}$ times the hash function on all the q' shared keys where $q' \geq q$. Since $\frac{p(k)}{p}$ is the probability that an established link is secured with k keys, the overall average number of hash computation is then:

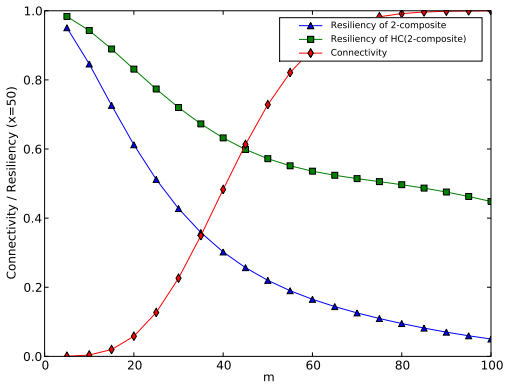
$$\sum_{k=q}^{m}(\frac{L^2-1}{6L} \times k \times \frac{p(k)}{p}) = \frac{L^2-1}{6L}\sum_{k=q}^{m}(k \times \frac{p(k)}{p})$$

□

In table 2, we summarize the average number of hash computations for $1 \leq L \leq 15$, when q=1, 2 and 3. Note that with $L = 1$ we have exactly the basic scheme where the pre-loaded keys are not hashed. This table shows that the average number of hash computations remains reasonable when $L \leq 15$.



(a) Connectivity



(b) Connectivity and resiliency (R_{50}) with q=2

Figure 5: Network connectivity and resiliency depending on the key ring size

Table 2: Average number of hash computation in HC(q-composite)

L	NH_{avg}	HC(q-composite)		
		q=1	q=2	q=3
1	0.0	0.0	0.0	0.0
2	0.25	0.493	0.659	0.861
3	0.444	0.877	1.171	1.531
4	0.625	1.233	1.646	2.152
5	0.8	1.578	2.107	2.755
6	0.972	1.918	2.561	3.348
7	1.143	2.255	3.01	3.936
8	1.313	2.589	3.457	4.52
9	1.481	2.923	3.902	5.102
10	1.65	3.255	4.346	5.682
11	1.818	3.587	4.789	6.261
12	1.986	3.918	5.232	6.84
13	2.154	4.249	5.673	7.417
14	2.321	4.58	6.115	7.994
15	2.489	4.91	6.556	8.571

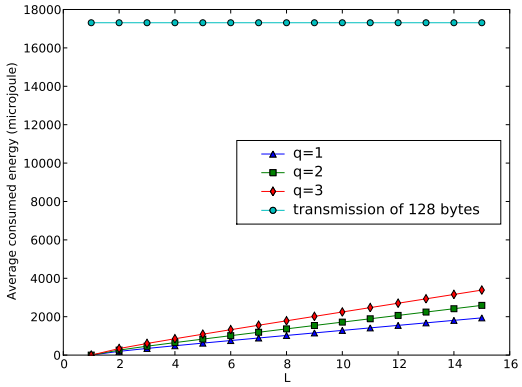


Figure 6: Energy consumption overhead of HC(q-composite)

Energy consumption overhead :
We were interested in evaluating the energy consumed due to the execution of hash functions over a mote's processor. For this end, we developed a nesC component of SHA1 hash function [23]. Then, we compiled the component for a Mica2 platform [24] using TinyOS [25]. We used the AVRORA simulator [26] to simulate and evaluate the consumed energy due to SHA1 calculation over a 160 bits string. The obtained results confirm our claims regarding the negligible induced energy overhead. Indeed, we found that the SHA1 calculations consume only 0.4 mJ. We also found that this is equivalent to only 4% of the energy consumed by a idle Atmel processor of a Mica2 node during 1 second. This means that 25 SHA1 executions consume only the equivalent energy required to maintain the node's processor idle during 1 second.

We have also evaluated the energy consumed by the radio module to send 128 bytes using 8×16 TOS messages. We found that the consumed energy is around 40 times the energy consumed by a hash function computation. We notice that this transmission takes about 600 milliseconds. In figure 6, we plot the average consumed energy by hash computations when using the HC(q-composite) scheme with q=1, 2 and 3. The figure shows that this energy is negligible compared to the energy consumed by the radio when sending 128 bytes. In addition, the hash function computation is done only one time during the common key discovery phase.

Choice of L's value:
As we have shown in the figures 4 and 5, our approach enhances the network resiliency. In addition, we notice from the proposition 2 that higher is L, better is the resilience. When L is too high, the value $\frac{L+1}{2L}$ approaches $\frac{1}{2}$, so the optimal resiliency with our approach is reached when $\frac{L+1}{2L}$ approaches $\frac{1}{2}$. However, we notice from proposition 3 that higher is L higher is the induced computational overhead. Indeed, the average number of hash computations depends linearly on the L value. We propose then to choose $L = 10$ which allows to approach the optimal resiliency ($\frac{L+1}{2L} = 0.55$ which is close to $1/2$), while the induced computational overhead is reasonable. For instance, with $L = 10$ the average hash computation is insignificant

(3.255 for q=1, 4.346 for q=2 and 5.682 for q=3).

5.2.4. Storage overhead

The storage memory required by the q-composite scheme is mainly the memory used to store the m pairs (key identifier, secret key) where the keys are selected randomly from S. If we neglect the q value storage memory, the required storage memory is then $M_{size} = m \times (K_{size} + I_{size})$ where K_{size} is the size of a given key and I_{size} is the size of the key identifier ($I_{size} = \log_2 |S|$ bits). Our solution requires exactly the same storage memory as the basic scheme in addition to the L value which is a short integer.

5.2.5. Communication overhead

The communication overhead depends linearly on the value of m. Indeed, each node must exchange the key identifiers and its own identifier in order to determine the common keys and then establish the secure link. The number of exchanged bits in the basic q-composite scheme is $m \times (\log_2 |S|)$ if we neglect the node identifier. As we maintain the same common key discovery phase with the basic scheme and which is based on the key identifier exchange, our scheme induces exactly the same communication overhead.

6. HC(SBIBD): A resilient combinatorial design-based key management scheme

In this section, we present a deterministic pool based key management scheme highly resilient against node capture. The latter results from the application of our hash chain class to the Symmetric Balanced Incomplete Block Design scheme proposed by Çamtepe and Yener [12]. First, we present the basic scheme, then we explain how to apply our hash chain class to this scheme, and finally we compare the performances of the two schemes.

In [12], Çamtepe and Yener introduced the use of combinatorial design for key pre-distribution in WSN. In order to guarantee the determinism when using a pool based key management scheme, authors made use of a particular combinatorial design which is the Symmetric Balanced Incomplete Block Design (SBIBD).

Given a finite set X of ν objects, a Balanced Incomplete Block Design (BIBD) is defined to be a set of k-distinct element subsets of X, called blocks, constructed is such a way that each object occurs in exactly r different blocks and every pair of distinct objects appears together in λ blocks. The number of resulting blocks is b. A BIBD (ν, b, r, k, λ) two properties: (i) $\lambda(\nu-1) = r(k-1)$ and (ii) $bk = \nu r$. It can then be identified by (ν, k, λ).
A BIBD is called symmetric when $b = v$ and in consequence $r = k$. A Symmetric BIBD (SBIBD) has the properties:

- every block contains k elements;
- each element occurs in exactly k blocks;
- each pair of elements occurs in λ blocks;
- each pair of blocks intersects in λ elements.

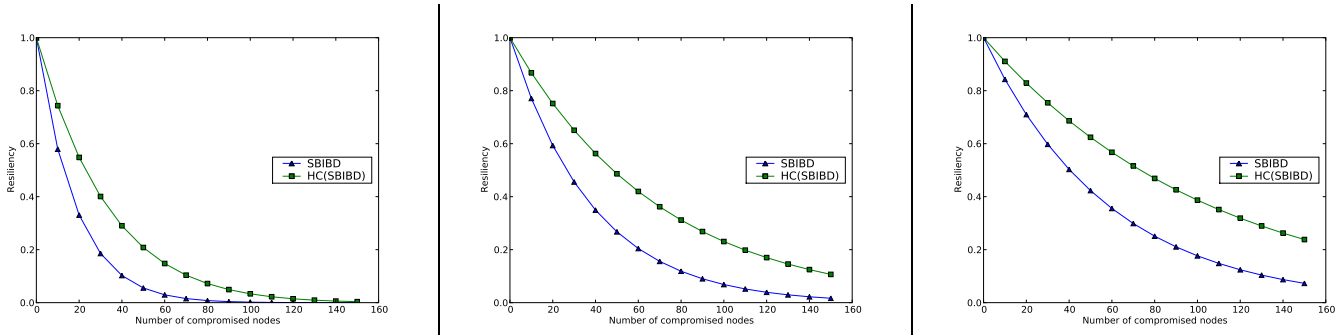
Table 3: Mapping from symmetric design to key distribution

SBIBD	Key distribution
X: Object set	S : Key pool
ν: Size of the object set X	$\|S\|$: Size of the key pool S
Blocks	Key rings
Objects in a Block (k)	Keys in a key ring (m)
number of generated blocks ($b = \nu = k^2 - k + 1$)	Number of generated key rings ($\|S\| = b = m^2 - m + 1$)
Two blocks share ($\lambda = 1$) object	Two key rings share ($\lambda = 1$) key

Let's take the example where we have 7 objects $\{1, 2, 3, 4, 5, 6, 7\}$ ($\nu = 7$). With ($k = 3$) and $\lambda = 1$, a configuration of the SBIBD blocks can be: $\{1, 2, 3\}\{1, 4, 5\}\{1, 6, 7\}\{2, 4, 6\}\{2, 5, 7\}\{3, 4, 7\}\{3, 5, 6\}$.
We have then 7 blocks ($b = \nu = 7$). Each block contains $k = 3$ distinct objects. Each object appears in $r = k = 3$ blocks, each pair of distinct elements occurs in $\lambda = 1$ block, and each pair of blocks intersects in exactly ($\lambda = 1$) object.

Çamtepe and Yener introduce in [12] a mapping from the SBIBD to the pool based key distribution. In this mapping, to each object is associated a distinct key, to the global set of objects is is associated the key pool and to each block is associated the node key ring (see table 3). We can then generate from a global key pool of $|S|$ keys, $|S|$ key rings of m keys in such a way that each two key rings shares exactly λ keys. Note that this generation has the property $\lambda(|S| - 1) = m(m - 1)$. Therefore, $|S| = m^2 - m + 1$ when $\lambda = 1$. In other terms, thanks to the symmetric design we can generate $m^2 - m + 1$ key subsets of m keys each one in such a way that each two key rings share exactly one common key. We use for that a global key pool S of $m^2 - m + 1$ keys. This approach guarantees a total connectivity coverage when nodes are pre-loaded with the generated subsets of keys.

The application of our hash chain-based approach to the SBIBD scheme gives birth to a deterministic key pre-distribution scheme which enhances the resiliency while ensuring a total connectivity coverage. Before the deployment step, $|S|$ key rings are generated from the key pool S where $|S| = m^2 - m + 1$. Instead of pre-loading each node with a distinct key ring, we propose to apply a hash function h, ($i \ mod \ L$) times on all the pre-loaded keys of each node, where i is the node identifier and L is a parameter of our class.
Thanks to the symmetric balanced incomplete block design (with $\lambda = 1$), any two neighboring nodes i and j share exactly one key having the same identifier. Let us consider s the identifier of the common key between the key rings of the neighboring nodes i and j. In the basic



(a) m=20 (b) m=40 (c) m=60

Figure 7: Network resiliency depending on the number of compromised nodes (SBIBD Vs HC(SBIBD))

scheme, the link secret key is $k_s = k_s^i = k_s^j$.
In our scheme HC(SBIBD), the node i is initially preloaded with $k_s^i = h^{i \ mod \ L}(k_s)$ where the node j is preloaded with $k_s^j = h^{j \ mod \ L}(k_s)$. The common secret key is then k_s^j if we suppose that $(i \ mod \ L) > (j \ mod \ L)$. Then, the node i computes its secret key with the node j as follows: $k_{ij} = h^{(i \ mod \ L)-(j \ mod \ L)}(K_s^i)$.

6.1. Analysis

In this section, We compare the performance of our scheme HC(SBIBD) against the basic SBIBD-based scheme with respect to the metrics defined in section 3.

6.1.1. Network resiliency against node capture

In the two schemes, each secure link is composed of one secret key. The fraction of compromised external links when x nodes are compromised is given as follows:

$$P(LC|NC_x) = \sum_{k=1}^{m^2-m+1} P(LC^k|NC_x)P(L_k) \qquad (6)$$

where:

- LC is the event that a link is compromised
- NC_x is the event that x nodes are compromised
- LC^k is the event that a link secured with the key having the identifier k is compromised.
- L_k is the event that a link is secured with a key having the identifier k.

Since keys having the identifier k occurs exactly in m key rings, the fraction of links secured with a key having the identifier k is then:

$$P(L_k) = \frac{\binom{m}{2}}{\binom{m^2-m+1}{2}} = \frac{1}{m^2-m+1} \qquad (7)$$

So, equation (6) becomes:

$$P(LC|NC_x) = \frac{1}{m^2-m+1} \sum_{k=1}^{m^2-m+1} P(LC^k|NC_x) \quad (8)$$

Proposition 4. *Using the basic SBIBD scheme, the probability that an external link is compromised when x nodes are compromised is:*
$P(LC|NC_x) = 1 - \prod_{t=0}^{x-1}(1 - \frac{m}{m^2-m+1-t})$

Proof. The probability that a link secured with the key having the identifier k is compromised when x nodes are compromised is the probability that the key k occurs one or more times in the x discovered key rings. Let us denote by $B_1, B_2, B_3, ...$ and B_x the x discovered key rings. Applying the bayes' theorem, we find that the probability that the key k occurs one or more times in the x discovered key rings is:

$$\begin{aligned} P(LC^k|NC_x) &= 1 - P(K_k \notin B_1 \cap .. \cap k_i \notin B_x) \\ &= 1 - P(K_k \notin B_1) \times \\ &\quad P(K_k \notin B_2 | K_k \notin B_1) \times ... \\ &\quad P(K_k \notin B_x | (K_k \notin B_1 \cap .. \cap K_k \notin B_{x-1})) \end{aligned}$$

Since the key k occurs exactly in m key rings from the m^2-m+1 possible key rings, we have:

$$\begin{aligned} P(LC^k|NC_x) &= 1 - ((1 - \frac{m}{m^2-m+1}) \times \\ &\quad (1 - \frac{m}{m^2-m+1-1}) \times ... \\ &\quad \times (1 - \frac{m}{m^2-m+1-(x-1)})) \\ &= 1 - \prod_{t=0}^{x-1}(1 - \frac{m}{m^2-m+1-t}) \end{aligned}$$

By replacing in equation (8), we find that the probability that an external link is compromised when x nodes are compromised is:
$P(LC|NC_x) = 1 - \prod_{t=0}^{x-1}(1 - \frac{m}{m^2-m+1-t})$

□

Table 4: Average hash computation in HC(SBIBD)

L	NH_{avg}	HC(SBIBD)
1	0.0	0.0
2	0.25	0.25
3	0.444	0.444
4	0.625	0.625
5	0.8	0.8
6	0.972	0.972
7	1.143	1.143
8	1.313	1.313
9	1.481	1.481
10	1.65	1.65
11	1.818	1.818
12	1.986	1.986
13	2.154	2.154
14	2.321	2.321
15	2.489	2.489

We give in appendix A another proof which is similar to the one given by Çamtepe and Yener in [12].

Proposition 5. *Using the HC(SBIBD) scheme, the probability that an external link is compromised when x nodes are compromised is:*
$P(LC|NC_x) = 1 - \prod_{t=0}^{x-1}(1 - \frac{L+1}{2L}\frac{m}{m^2-m+1-t})$

Proof. As proved in section 5.2.1, if a key having the identifier k is compromised, then the average fraction of discovered keys having the same identifier is $\frac{L+1}{2L}$ (the attacker discovers only derived versions). Similar to the proof of proposition 4 and following the same steps, we find that the probability that an external link is compromised when x nodes are compromised in the HC(BIBD) scheme is:
$P(LC|NC_x) = 1 - \prod_{t=0}^{x-1}(1 - \frac{L+1}{2L}\frac{m}{m^2-m+1-t})$
□

From propositions 4 and 5, we notice that our solution reduces the fraction of compromised links, the resiliency is so enhanced. We plot in figure 7 the resiliency of the two schemes, SBIBD and HC(SBIBD) depending on the number of compromised nodes ($R_x = 1 - P(LC|NC_x)$). We choose m respectively: 20 (a), 40 (b) and 60(c). Notice that the size of the global pool ($|S|$) is therefore 379, 1559 and 3539 respectively. We have chosen $L = 10$ when using our hash chain scheme. The figure shows that the network resiliency against node capture is better when using the HC(SBIBD) scheme. For instance, with $m = 40$ we enhance the resilience up to 20% when x is between 40 and 80.

6.1.2. Secure connectivity coverage

As explained before, the use of SBIBD theory ensures a total secure connectivity coverage. Each two nodes share exactly one key in their key subsets. When using our hash chain approach, we have also this property. The only difference is that the node having the low hash degree should apply the hash function a number of times on its shared key. Therefore, the two schemes ensure a total secure coverage (secure connectivity coverage = 1) thanks to the SBIBD based construction.

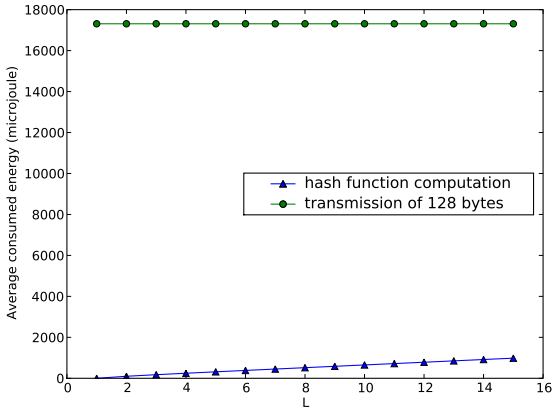


Figure 8: Energy consumption overhead of HC(SBIBD)

6.1.3. Computation overhead

After the common key discovery phase in the basic scheme, each two neighboring nodes determine their common shared key which is unique. This key is then the secret pairwise key used to secure the link.

With our solution, each two neighboring nodes determine, after the common key discovery phase, the identifier of their shared key. However, the node having the minimum hash degree should apply the hash function h a number of times to compute the common secret key. We have proved in lemma 2 that nodes have to apply in average $NH_{avg} = \frac{L^2-1}{6L}$ times the hash function on the their set of common keys before computing the secret key, where L is the parameter of our class. Using the SBIBD with $\lambda = 1$, each two nodes share exactly one key. Hence, nodes apply in average NH_{avg} times the hash function on the single common key. In the table 4, we compute the average number of hash computations when ($1 \leq L \leq 15$). Note that $L = 1$ represents the basic scheme because we do not apply at all the hash function. This table shows that the number of hash computations is negligible when $L \leq 15$. Since the energy consumption due to computation is negligible, we believe that the application of our solution to the SBIBD key management schemes does not degrade its performances.

In order to confirm our claim, we have evaluated the energy consumed by the hash function computation of HC(SBIBD) over a mote's processor. We simulated the computation of SHA1 hash function [23] over a Mica2 platform [24] using TinyOS [25]. We used the AVRORA simulator [26] to simulate and evaluate the consumed energy due to SHA1 calculation over a 160 bits string.

In figure 8, we plot the average consumed energy by hash computations when using the HC(SBIBD) scheme. This

figure shows that this energy is negligible compared to the energy consumed by the radio when sending 128 bytes. We notice also that the consumed energy in HC(SBIBD) is lower than the energy consumed by hash computations when applying HC to probabilistic schemes. Indeed, the hash function in HC(SBIBD) is applied on only one key during the common key discovery phase.

As explained before, we propose to choose $L = 10$ to achieve a good tradeoff between the resilience improvement and the computational overhead.

6.1.4. Storage overhead

We give the memory required to store the pairs (key identifier, secret key) in the basic SBIBD scheme as: $M_{size} = m \times (K_{size} + I_{size})$, where K_{size} is the size of a given key and I_{size} is the size of the key identifier: $I_{size} = \log_2 \left|m^2 - m + 1\right|$ bits. We summarize in table 5 the required storage memory when $K_{size} = 128$ *bits* (16 *bytes*) and when m is between 10 and 100. Notice that our scheme requires exactly the same storage memory in addition to the L value which is a short integer.

Table 5: Required storage memory in SBIBD & HC(SBIBD) schemes

m	I_{size} (bits)	I_{size} (bytes)	M_{size} (bytes)
10	7	1	170
20	9	2	360
30	10	2	540
40	11	2	720
50	12	2	900
60	12	2	1080
70	13	2	1260
80	13	2	1440
90	13	2	1620
100	14	2	1800

6.1.5. Communication overhead

The communication overhead depends linearly on the value of m. In fact, each node must exchange the key identifiers and its own identifier in order to find the only common key and then establish the secure link. The number of exchanged bits in the basic SBIBD scheme is then $m \times (\log_2 \left|m^2 - m + 1\right|)$ if we omit the node identifier. When applying our hash chain class, we maintain the same common key discovery phase with the basic scheme; our scheme induces then exactly the same communication overhead compared to the basic one.

7. Extensions and Enhancements

As shown above, the mechanism that we propose allows to enhance the resiliency of existing key pre-distribution schemes against oblivious node capture attacks. Nevertheless, the proposed scheme may incur unbalanced calculation overhead. Indeed, some nodes might hash their keys many times to deduce the shared key while others do not. Moreover, a smart attacker which captures nodes having the lowest values ($i \ mod \ L$) would allow a largest link compromise. In this section, we tackle these two issues and we propose countermeasures against both the smart attacks and the unbalanced overhead.

7.1. Smart node capture attacks:

The selective node capture attacks or smart attacks were proposed for the first time by Pietro et al. in [27]. Authors proposed in this paper a new attack where the adversary greedily uses the information disclosed by compromised nodes in order to choose the next node to capture. Authors defined a random variable called key information gain $G(s)$ for each potential node s. This variable is equal to the number of keys in the key ring of s which are not yet compromised. At each step, the attacker chooses the next node to compromise as the node which maximizes $E[G(s)|I(s)]$, the expectation of the key information gain $G(s)$ given the information $I(s)$ that the attacker knows on the key ring of sensor s. This approach allows maximizing the number of useful keys to collect at each step.

This kind of vulnerability is due mainly to the exchanges of key identifiers which may allow an attacker to know which keys are pre-loaded in which node. Therefore, countermeasures must hide key identifiers while allowing nodes to determine the shared keys. A lot of solutions were proposed. For instance, the *challenge response* solution introduced in [5] and used later in many schemes, consists of encrypting a challenge α with each key of a node. After receiving the encrypted values, the second node can deduce the shared keys by trying to decrypt the challenge α with its own keys. This solution is known to be secure but induces a high communication and computation complexity of order $O(m^2)$ where m is the key size. In *pseudo-random key index transformation*, used in [28] for instance, each node generates the key indexes using a pseudo-random generator initialized with a given node seed. Even this solution is communication efficient, it is not secure since the attacker can generate key indexes knowing the pseudo-random generator and the seeds which are usually supposed to be public. Authors in [19] propose to enhance the Çamtepe scheme in order to avoid key identifier exchanges and counter active attacks. For this purpose, they index all nodes and keys and propose a mapping between node indexes and key indexes. This solution remain efficient as long as the mapping algorithm is not known, otherwise, an attacker can easily reveal the key identifers knowing the node identifier.

Pietro et al. proposed in [27][29] an efficient and secure shared key discovery way trading off communications for local computations. In the proposed solution, each node i is pre-loaded with keys from the global pool such as : $f_y(i \| K_u) \equiv 0 \ mod \ (\frac{m}{S})$ where f_y is a one way pseudo-random function having y as a seed. So, each node can deduce the shared keys with a neighbor by verifying the relation on its own keys.

When using our hash function based mechanism, all the proposed countermeasures, presented above, can be

adapted to face this kind of attacks. In addition, a particular smart attack can be launched and should be considered: the attacker may target nodes having lower hash degree ($i \ mod \ L$) in order to compute a larger number of derived keys and hence compromise more links. In the worst case, the attacker could target only the nodes having the identifier i such that ($i \ mod \ L$) = 0, in this case the attacker will have original keys which can be used to compute derived versions and so nullifies the effect of our solution.

7.2. Unbalanced calculation overhead

We proved in previous sections that the average introduced computational overhead, due to hash calculations, is insignificant. Nevertheless, it is obvious that the limited introduced workload is unbalanced. Indeed, most calculations are shifted to nodes having a low hash degree ($i \ mod \ L$), , where i is the node identifier and L is the parameter of our approach. Using the proposed mechanism HC, a node i have to perform hash computations with a probability $\frac{L-1-(i \ mod \ L)}{L}$ which is unbalanced.

7.3. Balanced and smart-attack free extension

We propose in this subsection an improvement of our hash-based mechanism which alleviates both the smart attack presented above and the computation overhead unbalance. For analysis simplicity, we assume that the key ring size m is an even number, we assume also that the key identifiers are totally ordered and that the keys within a key ring are distinct.

In order to balance hash computations and render the above smart attack inefficient, we divide the original keys, to be pre-loaded in each node i, to two subsets of size $\frac{m}{2}$ each. The first subset contains the keys having the lower identifers while the second one contains the keys having the higher identifers. Before de deployment, we propose to apply a hash function h, ($i \ mod \ L$) times to the keys of the lower subset and $((L-1)-(i \ mod \ L))$ times to the keys of the higher subset. Thus, each node i is preloaded with the set of keys KR^i such as:
$KR^i = LKR^i \cup HKR^i$ where:
$LKR^i = \left\{h^{i \ mod \ L}(K_1), ..., h^{i \ mod \ L}(K_{\frac{m}{2}})\right\}$ and
$HKR^i = \left\{h^{L-1-(i \ mod \ L)}(K_{\frac{m}{2}+1}), ..., h^{L-1-(i \ mod \ L)}(K_m)\right\}$
where $K_1, K_2, ...K_m$ are the distinct original keys to be pre-loaded in node i totaly ordered with respect to their identifiers.

After the deployment step, each node can determine the hash degree of a shared key thanks to the common key discovery phase. Hence, each node can deduce whether it applies the hash function on the stored key or it uses this key without hash computation.

The use of this enhanced version allows to balance the hash function computations between nodes. Indeed, for any given two nodes i and j, the node i will perform the hash function with probability $\frac{1}{2}$ if we assume that any given shared key belongs to the lower set or the higher subset with the same probability $\frac{1}{2}$.

Let us assume that s is the identifier of a given shared key and that ($i \ mod \ L$) $\neq$ ($j \ mod \ L$) and that ($i \ mod \ L$) $\neq$ $L-1-(j \ mod \ L)$. The probability that the node i has to apply the hash function is then given by :

$$\begin{aligned}
P &= P(K_s \in LKR^i \cap K_s \in LKR^j) \times \\
&\quad P(i \ mod \ L < j \ mod \ L) \\
&+ P(K_s \in LKR^i \cap K_s \in HKR^j) \times \\
&\quad P(i \ mod \ L < L-1-(j \ mod \ L)) \\
&+ P(K_s \in HKR^i \cap K_s \in LKR^j) \times \\
&\quad P(L-1-(i \ mod \ L) < j \ mod \ L) \\
&+ P(K_s \in HKR^i \cap K_s \in HKR^j) \times \\
&\quad P(L-1-(i \ mod \ L) < L-1-(j \ mod \ L)) \\
&= \frac{1}{4} \times [P(i \ mod \ L < j \ mod \ L) + \\
&\quad P(L-1-(i \ mod \ L) < L-1-(j \ mod \ L))] \\
&+ \frac{1}{4} \times [P(i \ mod \ L < L-1-(j \ mod \ L)) + \\
&\quad P(L-1-(i \ mod \ L) < j \ mod \ L)] \\
&= \frac{1}{2}
\end{aligned}$$

Following a similar reasoning, we find that when ($i \ mod \ L$) = ($j \ mod \ L$) or ($i \ mod \ L$) = $L-1-(j \ mod \ L)$ exclusively, the probability that the node i has to apply the hash function is also equal to $\frac{1}{2}$. Finally, if ($i \ mod \ L$) = ($j \ mod \ L$) and ($i \ mod \ L$) = $L-1-(j \ mod \ L)$, both nodes do not perform hash computations.

In addition to load balancing hash computations, the proposed enhancement allows to counteract the smart attack presented above. Indeed, when using the basic solution, the probability that a key is compromised following the capture of a node i is $\frac{L-(i \ mod \ L)}{L}\frac{m}{|S|}$. In other words, the capture of nodes having low values ($i \ mod \ L$) reveals a high number of pre-loaded keys while the capture of nodes having high values ($i \ mod \ L$) reveals low number of keys. The above smart attack relies on this unbalanced revealing of keys following a node capture.

When using the enhanced version presented in this subsection, the probability that a given key is compromised following the capture of a node i is equal to $\frac{L-(i \ mod \ L)}{L} \times \frac{m}{2|S|}$ due to the capture of the lower subset plus $\frac{L-[(L-1)-(i \ mod \ L)]}{L} \times \frac{m}{2|S|}$ due the capture of the highest subset (the two subsets within a key ring are assumed to be distinct). Therefore, the capture of one node reveals a fraction of $\frac{L+1}{2L}\frac{m}{|S|}$ of keys independently of its identifier. Hence, the smart attack described above can no more be conducted against this enhanced scheme, since the number of keys revealed following a node capture is the same whatever the identifier of the node.

8. Conclusion

Asymmetric key management schemes are likely to be unsuitable for WSN because of resource limitations. Moreover, because of the lack of infrastructure in WSN, it is difficult to assume the existence of a trusted third party which can attribute pairwise secret keys to neighboring nodes. This is why symmetric key pre-distribution schemes are the most suitable paradigm for wireless sensor networks to secure exchanges.

In this paper, we presented a hash chain class of key management schemes highly resilient against node capture. We enhance resilience by concealing keys through the use of an efficient hash chaining mechanism. Our class can be applied to any pool based key pre-distribution scheme. In this work, we applied it to the well known probabilistic q-composite scheme in a first time and to the deterministic Symmetric Balanced Incomplete Bock Design scheme in a second time. The analytical study showed that our approach may enhance resiliency up to 40% without introducing any new storage or communication overheads, it induces an insignificant computational overhead.

9. Acknowledgement

This work is made as part of the Picardie regional project under reference I159C. The authors wish to thank the Picardie regional council in France and the European Regional Development Fund (ERDF) for funding and supporting this project.

Appendix A. proof of proposition 4

When using the SBIBD scheme, the probability that a link secured with the key having the identifier k is compromised when x nodes are compromised is the probability that the key k occurs one or more times in the x discovered key rings. Since the key occurs exactly in m key rings among the possible $m^2 - m + 1$ key rings, we have:

$$\begin{aligned}
P(LC^k|NC_x) &= 1 - \frac{\binom{(m^2-m+1-m)}{x}}{\binom{m^2-m+1}{x}} \\
&= 1 - \frac{\binom{m^2-2m+1}{x}}{\binom{m^2-m+1}{x}} \\
&= 1 - \frac{(m^2-2m+1)!}{(m^2-m+1)!}\frac{(m^2-m+1-x)!}{(m^-2m+1-x)!} \\
&= 1 - \prod_{t=0}^{x-1} \frac{m^2-2m+1-t}{m^2-m+1-t} \\
&= 1 - \prod_{t=0}^{x-1} (1 - \frac{m}{m^2-m+1-t})
\end{aligned}$$

Following the equation (8), we find that the probability that an external link is compromised when x nodes are compromised is:

$$P(LC|NC_x) = 1 - \prod_{t=0}^{x-1} (1 - \frac{m}{m^2-m+1-t})$$

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.

[2] R. Watro, D. Kong, S. Cuti, C. Gardiner, C. Lynn, and P. Kruus. Tinypk: securing sensor networks with public key technology. In *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, pages 59–64. ACM Press, 2004.

[3] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Cryptographic Hardware and Embedded Systems (CHES)*, pages 119–132, 2004.

[4] J. Zhang and V. Varadharajan. Wireless sensor network key management survey and taxonomy. *Journal of Network and Computer Applications*, 33(2):63 – 75, 2010.

[5] L. Eschenauer and V.D. Gligor. A key-management scheme for distributed sensor networks. In *ACM CCS '02*, pages 41–47, 2002.

[6] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE SP '03*, 2003.

[7] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *IEEE INFOCOM*, pages 586–597, 2004.

[8] C. Castelluccia and A. Spognardi. A Robust Key Predistribution Protocol for Multi-Phase Wireless Sensor Networks. In *IEEE Securecom*, pages 351–360, 2007.

[9] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *ACM CCS*, pages 52–61, 2003.

[10] Z. Yu and Y. Guan. A robust group-based key management scheme for wireless sensor networks. In *Wireless Communications and Networking Conference*, pages 1915–1920. IEEE, 2005.

[11] S. Zhu, S. Setia, and S. Jajodia. Leap: efficient security mechanisms for large-scale distributed sensor networks. In *ACM CCS*, pages 62–72, 2003.

[12] S. A. Çamtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 15:346–358, April 2007.

[13] B. Maala, Y. Challal, and A. Bouabdallah. Hero: Hierarchcal key management protocol for heterogeneous wsn. In Springer Boston, editor, *IFIP WSAN*, pages 125–136, 2008.

[14] K. Kifayat, M. Merabti, Q. Shi, and D. Llewellyn-Jones. Security in wireless sensor networks. In *Handbook of Information and Communication Security*, pages 513–552. 2010.

[15] D. Sánchez and H. Baldus. Key management for mobile sensor networks. In *Secure Mobile Ad-hoc Networks and Sensors*, volume 4074 of *Lecture Notes in Computer Science*, pages 14–26. Springer Berlin / Heidelberg, 2006.

[16] A. Ouadjaout, M. Bagaa, A. Bachir, Y. Challal, N. Lasla, and L. Khelladi. *Encyclopedia on Ad Hoc and Ubiquitous Computing*, chapter Information Security in Wireless Sensor Networks, pages 427–472. World Scientific, 2009.

[17] C. Blundo, A. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, pages 471–486. Springer-Verlag, 1993.

[18] R. Blom. An optimal class of symmetric key generation systems. In *Proceedings of the Eurocrypt 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques.*, pages 335–338. Springer-Verlag, 1985.

[19] S. Ruj and B. Roy. Key predistribution using combinatorial designs for grid-group deployment scheme in wireless sensor networks. *Transactions On Sensor Networks*, 6(1), 2009.

[20] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. Spins: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.

[21] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar. Spins: security protocols for sensor netowrks. In *ACM MOBICOM*, pages 189–199, 2001.

[22] W. Bechkit, A. Bouabdallah, and Y. Challal. Enhancing resilience of probabilistic key pre-distribution schemes for wsns through hash chaining. In *ACM CCS*, pages 642–644, 2010.

[23] National Institute of Standards and Technology. Secure hash standard. *Federal Information Processing Standards Publication*, 1995.

[24] http://www.memsic.com.

[25] http://www.tinyos.net.

[26] B. L. Titzer, D. K. Lee, and J. Palsberg. Avrora: Scalable sensor network simulation with precise timing. In *IPSN*, 2005.

[27] R. Di Pietro, L. V. Mancini, and A. Mei. Efficient and resilient key discovery based on pseudo-random key pre-deployment. *Parallel and Distributed Processing Symposium, International (IPDPS)*, 13:217b, 2004.

[28] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In *IEEE International Conference on Network Protocols*, pages 326–, 2003.

[29] R. Di Pietro, L. V. Mancini, and A. Mei. Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks. *Wireless Networks*, 12(6):709–721, 2006.