HERO: Hierarchcal key management protocol for heterogeneous wireless sensor networks

Boushra Maala, Yacine Challal, Abdelmadjid Bouabdallah

HERO: Hierarchical kEy management pRotocol for heterOgeneous wireless sensor networks

Boushra Maala and Yacine Challal and Abdelmadjid Bouabdallah

Universite de Technologie de Compiegne, UMR CNRS 6599 Heudiasyc. France
(balkubei, ychallal, bouabdal)@hds.utc.fr

Abstract. Early researches focused on the security of homogenous sensor networks. However, recent works have demonstrated that the presence of heterogeneous sensor nodes gives better performance than homogenous ones in terms of energy consumptions, storage overhead, and network connectivity. In this paper, we propose a hierarchical key management scheme named HERO which is based on random key pre-distribution. HERO aims to construct a secure tree instead of complete connected graph as in existing schemes. Thanks to this realistic assumption, our key management scheme reduces considerably the number of pre-loaded keys assigned to each node while maintaining high security level at the same time. The preliminary simulation results using TOSSIM demonstrates that our scheme outperforms existing ones with respect to storage overhead.

1 Introduction

Recent advances in Wireless Sensor Networks (WSN) technology make them ideal candidates to various applications [5, 7] such as environment monitoring, military surveillance, forest fire monitoring, agriculture monitoring, etc. Security in WSN is a critical challenge especially for uncontrolled and hostile environments. Indeed, security vulnerabilities in WSN stem from their characteristics [8] that make them attractive: the wireless nature of radio channels exposes transmitted data in WSN to eavesdropping and manipulation, the resources limitations of motes prevent using heavy cryptographic techniques, and the deployment of sensors, generally, in unattended areas exposes them to intentional and accidental physical destruction. However, recent researches show that the performance and the lifetime of WSN can be improved by using heterogeneous sensor nodes instead of homogenous ones without significantly increasing the cost [9]. In Heterogeneous Sensor Networks (HSN) there is a few powerful and expensive nodes, and a large number of inexpensive and simple sensor nodes. A few key management schemes [11, 3, 4] have been proposed for (HSN). All these schemes try to exploit the heterogeneity features, in order to reduce the number of stored keys per node, and to guarantee a good level of security at the same time.
In this paper, we present an effective key management scheme based on probabilistic key pre-distribution; HERO(Hierarchical kEy management pRotocol for heterOgeneous wireless sensor networks). HERO provides a key management solution to secure a HSN relying on the construction of a secure tree. Moreover, it reduces the number of loaded keys because it needs only to store the keys which are necessary to build the secure tree, not to secure all possible links in the network as in all existing schemes[2, 11,

Please use the following format when citing this chapter:

Maala, B., Challal, Y. and Bouabdallah, A., 2008, in IFIP International Federation for Information Processing, Volume 264; Wireless Sensor and Actor Networks II; Ali Miri; (Boston: Springer), pp. 125–136.

3, 4]. The rest of the paper is organized as follows. In section II, we report on the related works presented in the literature. In section III, we detail our scheme. We evaluate the performances of HERO in section IV. In section V, we present some simulation results. Finally, we conclude the paper in section VI.

2 Related Works

Most of previous key management solutions have been proposed for homogeneous WSN, while a few works have been recently proposed for heterogeneous ones [11, 3]. The majority of these solutions are based on probabilistic key-distribution proposed by Eschenauer and Gilgor in [2]. This scheme is based on probabilistic key sharing among the sensor nodes. Prior to sensor network deployment, a large pool of P keys and their identifiers are generated. Each sensor node is pre-loaded with a key ring of k keys and their associated key identifiers which are selected randomly from the key pool without replacement. After sensor network deployment, each two neighbor sensor nodes compare their list of key identifiers in order to discover their shared key which will be used to secure the link between them. Due to the random distribution of keys to each sensor node, a shared key may be not available between two nodes. In this case, it is necessary to find a secure path composed of secure links using the shared keys of intermediary nodes. The probability of secure path existence using this scheme depends on the number of pre-loaded keys in each sensor. Unfortunately, to reach high probabilities this scheme may induce high storage overhead not supported by many types of sensor nodes. New key management schemes have been proposed for (HSN). These schemes are based on the basic scheme of probabilistic key pre-distribution [2]. Du et al. [11] proposed the Asymmetric Pre-distribution (AP) key management scheme. Its main idea is to pre-load a relatively large number of keys in each one of a small number of powerful nodes (H-sensor), while only a small number of keys are stored in each one of nodes (L-sensor) which have very limited space of storage and capacity of communication. Due to the usage of these two types of nodes, two different types of key rings have been used to achieve a high probability that each two nodes share at least one shared key. Indeed, AP scheme is more scalable than the basic scheme and it reduces the number of pre-loaded keys compared to the basic scheme, but it is still sometimes unsuitable for some types of sensors due to memory constraints. Also, in order to reduce the storage requirements and to maintain the same security strength of the basic scheme and AP scheme [2, 11], Hussain et al. [3] proposed a key generation process to reduce the key storage requirement. Brown et al. in [4], proposed a public-key-based scheme. In order to establish a secure communication topology in the network, each H-sensor needs to store (4+DH) keys where DH corresponds to the degree of this H-sensor. This means that each H-sensor stores enough keys to communicate with each node within its cluster, and the number 4 refers to four types of keys which are its public key, its private key, its group key, and the public key of the base station. For L-sensors, each node u must store (3+2Du) where Du corresponds to the group keys for all its neighbors, and the number 3 refers to three types of keys: its public key, its private key, and the public key of the base station. Although, this scheme may not need too large storage compared to other schemes, it is based on public key cryptography which is too heavy for tiny

sensors. Recently, Traynor et al. [12] presented a protocol named LIGER. This protocol is based on the presence of a key distribution center(KDC), which stores the mapping of pre-loaded keys to nodes. This protocol consists of two components: the first is a protocol for a network in an infrastructure-less (i.e. without KDC). This component is defined as a protocol to instantiate probabilistic keying, and is referred to as LION. While, the second component is relied to the availability of KDC (i.e. network with KDC). It exploits the knowledge of the pre-loaded keys to perform the probabilistic authentication with a high degree of confidence. In addition, session keys are established with enforcement of the least privilege. This component is referred to as TIGER

All the schemes presented above assume the requirement to find a shared key between any two nodes in the network. We believe that this assumption is too strong. In fact, the primary goal in most WSN applications is to collect sensed information from the observed area, and not to have a complete connected graph. This can be achieved through a many-to-one communication scheme. Thus, we propose in our scheme HERO to elevate this assumption while achieving this primary goal. In HERO we replace the construction of a completely connected secure graph with a secure tree. As we can see in the performance evaluation sections, this allows to considerably reduce the required number of preloaded keys.

3 HERO(Hierarchical kEy management pRotocol for heterOgeneous wireless sensor networks) description

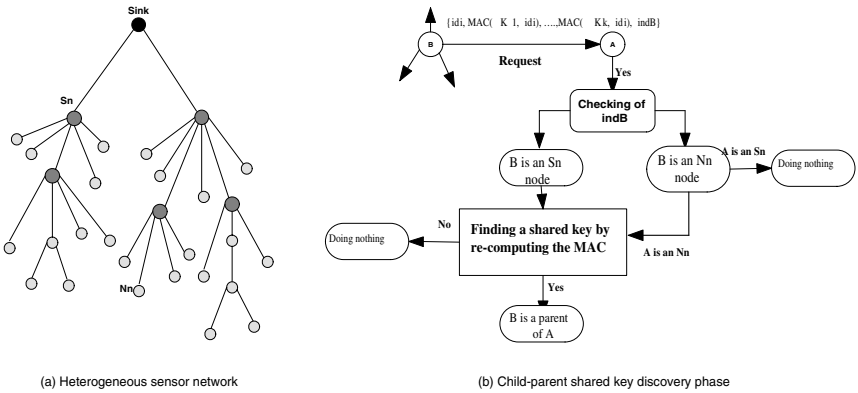
HERO is a new key management scheme that allows to create a secure tree rooted at the sink. Limiting the key sharing requirement to the child-parent relationship allows to reduce the key storage overhead to few keys per mote, as we can see in the performance evaluation section.
In HERO, we adopt a heterogeneous sensor network (HSN) consisting of a powerful base station (the Sink), and two types of sensor nodes: a small number of powerful sensor nodes with a large radio range (r), which are referred as Super nodes (Sn), and a large number of simple nodes with a small radio range (μ), which are referred as Normal nodes (Nn). Since sensor type influences the tree construction, each mote embeds to its sent messages an indicator (ind) that specifies the mote category (Sn or Nn). In the following paragraphs, we describe our scheme in details.

3.1 HERO Details

HERO scheme includes essentially two phases: Key pre-distribution and child-parent shared key discovery. In addition, we discuss two important operations in WSN: establishing keys for newly deployed sensors and the revocation of compromised keys.

Key Pre-distribution A large key pool (P) of P keys and the corresponding key Ids are generated off-line. Then, each Sn node is pre-loaded with K keys and their corresponding key Ids. These keys are randomly selected from the large key pool without replacement. We refer to this set of keys as KP (Key Pool), such that $KP_A \cap KP_B \neq \phi$ where A

and B are two Sn nodes. Then each Nn node is pre-loaded with m ($m \ll K$) keys and their corresponding key Ids. These keys are randomly selected from P without replacement. We refer to this set of keys as PKP (Parcel Key Pool), such that $PKP_i \cap PKP_j \neq \phi$, and $PKP_i \cap KP_X \neq \phi$ where i and j are two Nn nodes and X is a Sn node.



(a) Heterogeneous sensor network

(b) Child-parent shared key discovery phase

Fig. 1. Construction of a secure tree

Child-parent Shared Key Discovery The goal of HERO is to construct a secure tree (see figure 1). Therefore, HERO aims to construct a secure path between each node and the Sink, contrary to other schemes [11, 3] that aim to build a secure complete graph. After pre-loading each node with its key ring as described above, the Sink broadcasts a tree construction request. Since the Sink stores the key pool P, it is sure that all sensor nodes have at least one shared key with the Sink. When a sensor node (Sn or Nn) receives this request from the Sink, it considers the Sink as its parent, and selects one key of its key ring as its shared key with the Sink. These nodes are now attached to the tree rooted at the sink. Each newly attached node forwards the " tree construction request " downstream as follows:
Each attached node i (Sn or Nn) authenticates the request through computing one *Message Authentication Code* (MAC), over its identifier, using each key of its key ring. For instance, the "tree construction request" forwarded by node i will be:
$\{id_i, MAC(K1, id_i), MAC(K2, id_i), \ldots, MAC(Kk, id_i), ind_i\}$, where k is the size of node's key ring and ind_i refers to the type of node i. Then, node i broadcasts the request. When a node receives this message, it checks the type of the node. Thus, there are two different cases:

1. Receiving node is an Sn node:
   Suppose that A is the receiving node and B is the sending node. Then, A checks the type of node B. So:
   - If B is an Sn node. Then A checks the request message in order to discover if it has a shared key with B or not. This is done by verifying the corresponding

MAC. Here, if A finds a shared key with B, it notes node B as its parent and notes their shared key in its table of routing. Otherwise, A does nothing and it waits receiving a request from another node.
- If B is an Nn node. Then A waits wishing receiving a request from an Sn node.

2. Receiving node is an Nn node: If A finds a shared key with B, it notes B as its parent in its routing table and their shared key. Otherwise, it waits receiving another request.

Establishing keys for newly deployed sensors The deployment of new nodes in WSN may be required for several reasons such as: Solving the problem of the miss of coverage in the tested area, enhancing the accuracy of the sensed phenomena, and prolonging the lifetime of the deployed network.
To attach a new node A to the secure tree, the new node must search for a parent, already attached to the secure tree, with whom it shares a key. Thus, A is firstly pre-loaded with its key ring of k keys which are selected randomly from the key pool. Then, it broadcasts a request message as follows:
$\{Id_A, MAC(K1, id_A), MAC(K2, id_A), \ldots, MAC(Kk, id_A), ind_A\}$.
When a deployed sensor node B, which is already an attached node, receives this message, it broadcasts the following message:
$\{Id_B, MAC(K1, id_B), MAC(K2, id_B), \ldots, MAC(Kk, id_B), ind_B\}$.
When A receives this message, it checks if it has a shared key with B. If A and B have a shared key, A notes B as its parent. Otherwise A waits receiving a message from another node. Furthermore, due to one of the reasons which are cited above, a deployed node C may be still unattached to the secure tree. So, upon receiving the request of the new node A, C attempts to profit from this new node to attach to the secure tree. Thus, it checks if there is a shared key with A, if a shared key is found, C considers A as its parent.

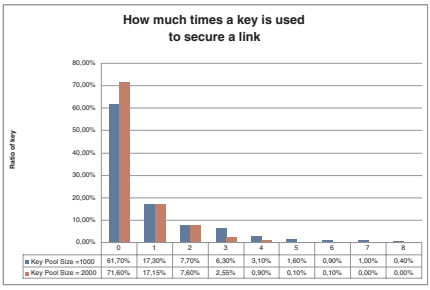


Fig. 2. Frequency of key usage with different key pool sizes

Key Revocation Whenever a sensor node is compromised, it is necessary to revoke all its keys. First, the sink generates a signature key that it sends securely to each sensor

except the compromised one. One method to distribute this signature key securely to the sensors [2], is to encrypt it with a list ok keys selected from the key rings of not compromised sensors, so that all sensors could decrypt the signature key except the compromised one. Then the sink broadcasts a message containing the identifiers of the compromised keys authenticated with the signature key. Upon receiving this revocation message, each sensor node verifiers the signature using the signature key. Then, it removes the corresponding keys from its key ring. However, due to the revocation of the compromised keys, some links may disappear. So, in order to reconfigure these links; the affected nodes restart a new child-parent shared key discovery phase.
Figure 2 shows that, out of P=1000 keys, only 60% of the keys are used to secure links, only 17.30% are used to secure one link, 7.70% are used to secure two links, and only 6.30% are used to secure three links. This means that compromising one key leads to the compromise of another link with probability 0.17, of two other links with probability 0.07 and so on. Moreover, the compromise of one Nn node of key ring size m=5 leads to reconfigure only 4 links in average. So, a very small number of nodes are affected due to the compromise an Nn. Therefore, only a small number of messages are required to reconfigure the affected links.

4 Evaluation of HERO

In order to construct a secure tree with a minimum key storage overhead, we must find the minimum size of an Nn's key ring such that the probability of that Nn attaches to the secure tree be high. An Nn node attaches to the secure tree as a child of an Sn node except when this Nn node has not any shared key with at least one Sn node within its neighborhood. In this case, as we have explained in previous section, the Nn node attaches to the secure tree as a child of another Nn node within its neighborhood with which it has a common key. Therefore, the probability that an Nn node attaches to the secure tree (P_a) is the probability that an Nn node shares at least one key with at least one Sn within its neighborhood (P_{Sn}) or that it does not share any key with any Sn node within its neighborhood ($1 - P_{Sn}$) and it shares at least one key with at least one Nn within its neighborhood (P_{Nn}). Thus, we can write this as:

$$P_a = P_{Sn} + (1 - P_{Sn}) \times P_{Nn} \tag{1}$$

Where, P_{Sn} is the probability that an Nn node shares at least one key with at least one Sn of its neighborhood, then P_{Sn} = 1- the probability that no key in common between an Nn node and any one of Sn node within its neighborhood. Then,

$$P_{Sn} = 1 - \left[\frac{\binom{P}{K+m}\binom{K+m}{m}}{\binom{P}{m}\binom{P}{K}} \right]^{N_{Sn}} \tag{2}$$

where N_{Sn} is the number of Sn nodes within a Nn node's neighborhood.
Similarly, since P_{Nn} is the probability that an Nn node shares at least one key with at least one Nn within its neighborhood, then, P_{Nn} =1- the probability that no key in

common between an Nn node and any one of Nn node within its neighborhood. Then,

$$P_{Nn} = 1 - \left[\frac{\binom{P}{2\times q}\binom{2\times q}{q}}{\binom{P}{m}^2} \right]^{N_{Nn}} \tag{3}$$

where N_{Nn} is the number of Nn nodes within a Nn node's neighborhood. Thus, from 1, 2 and 3, and by using *Stirling approximation* $n! \simeq \sqrt{2\pi n}.(\frac{n}{e})^n$, we obtain:

$$P_a \simeq H^{N_{Sn}} + V^{N_{Sn}} \times U^{N_{Nn}} \tag{4}$$

where:

$$H = 1 - \sqrt{1 + \frac{m\times K}{P^2 - P\times(m+K)}} \times \left(1 + \frac{m\times K}{P^2 - P\times(m+K)}\right)^P \times \left(1 - \frac{K}{P-m}\right)^m \times \left(1 - \frac{m}{P-K}\right)^K$$

$$V = \sqrt{1 + \frac{m\times K}{P^2 - P\times(m+K)}} \times \left(1 + \frac{m.K}{P^2 - P\times(m+K)}\right)^P \times \left(1 - \frac{K}{P-m}\right)^m \times \left(1 - \frac{m}{P-K}\right)^K$$

and

$$U = 1 - \frac{(1-\frac{m}{P})^{2(P-m+1/2)}}{(1-\frac{2m}{P})^{(P-2m+1/2)}}$$

Let us assume that $N = n_1 + n_2$ (n_1 and n_2 being the numbers of Sn and Nn nodes respectively, where $n_1 << n_2$) are uniformly distributed within a square area having a surface Z=$L \times L\ m^2$. Thus, the density of deployed Sn and Nn nodes are $D_{Sn} = \frac{n_1}{Z}$ and $D_{Nn} = \frac{n_2}{Z}$ respectively. Then, the possible number of Sn node within a zone which is covered by an Nn node is $N_{Sn} = D_{Sn} \times Z_{Nn} = \frac{n_1\times\pi\times\mu^2}{Z}$, where μ is the radius of Nn node. Also, the possible number of Nn nodes within a zone which is covered by an Nn node is $N_{Nn} = D_{Nn} \times Z_{Nn} = \frac{n_2\times\pi\times\mu^2}{Z}$.

Hereafter, we present an example of a network consisting of 10 IRIS motes as Sn nodes with radio range r=350m, and 1000 MicaZ motes as Nn nodes with radio range μ=90m. These sensor nodes are deployed uniformly within a square-grid of $500 \times 500m^2$ with density $D = \frac{N}{Z} = 40.4 \times 10^{-4}$ node/m^2. Then, the possible number of Sn and Nn nodes within an Nn node's neighborhood are: N_{Sn}=1 node and N_{Nn}= 102 nodes respectively.

By using the equation 4, where P=3000, m=5 and K=100, we obtain Pa=0.6397. Thus, we notice that an Nn node needs only to be pre-loaded with a very small number of keys to be attached to the secure tree with a relatively high probability.

Figure 3 shows the probability of attachment Pa as function of m and K i.e. Pa=f(m,K) where m=$\{1,2,3,4,5\}$ and K=$\{100,150,200,250,300\}$.

Figure 3.a illustrates Pa=f(m,K) for P=1000, while figure 3.b illustrates Pa=f(m,k) for P=3000. However, by comparing these two figures, we can clearly notice that Pa decreases when P increases when m and K are fixe. For instance, with K=100 and m=3 we obtain Pa= 0.7097 for P=1000 and 0.335 for P=3000. Moreover, Pa increases when K increases when m and P are fixe. For example, with P=1000 and m=1, there

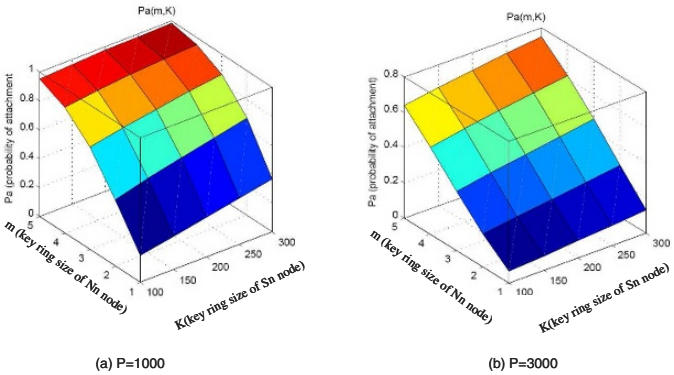


Fig. 3. Pa=f(m,K) where K={100,150,200,250,300}, m={1,2,3,4,5}, $N_{Sn} = 1$, N_{Nn}=102

is an increment of 24% in Pa when K increases from 100 to 150. We notice also that Pa increases when m increases when K and P are fixe. For instance, for P=1000 and K=100, there is an increment of 54% in Pa when m increases from 2 to 3. Thus, we can conclude that with a very small size of m, HERO can give a high probability of attachment, and hence a high probability of constructing a secure tree. HERO can also be considered as an efficient protocol since it provides a high security level (compared to existing protocols) with a very small key storage overhead.

5 Simulation and analysis

We have implemented a prototype of HERO using TinyOS [6] environment. TinyOS is an event-driven operating system commonly used on WSN motes. The programs are written in nesC which is a C-like language [10, 14]. Basically, we have defined a new interface KeyManager, and developed a module implementing this interface KeyManagerM. It contains all commands and events which are necessary to manage the keys in a network. We have also developed the component TreebuilderM that uses KeyMangerM to build the secure tree.

In what relates to the network topology, we considered a square-grid of $500 \times 500m^2$. Super nodes have a 350m radio range (like IRIS motes), and Normal nodes have a 90m radio range (like MicaZ motes) [13]. The positions of the nodes are distributed uniformly over the square grid. We assumed also that 1% of the nodes are Super nodes. We carried out intensive simulations of our prototype using TOSSIM [1] which emulates the execution of the developed code over a personal computer. We were mainly interested in evaluating the secure coverage ratio with respect to the network size, and the key ring size at Super Nodes and Normal Nodes for different key pool sizes. We mean by the secure coverage ratio the number of motes that succeed to establish a secure path to the Sink, over the total number of motes in the network.

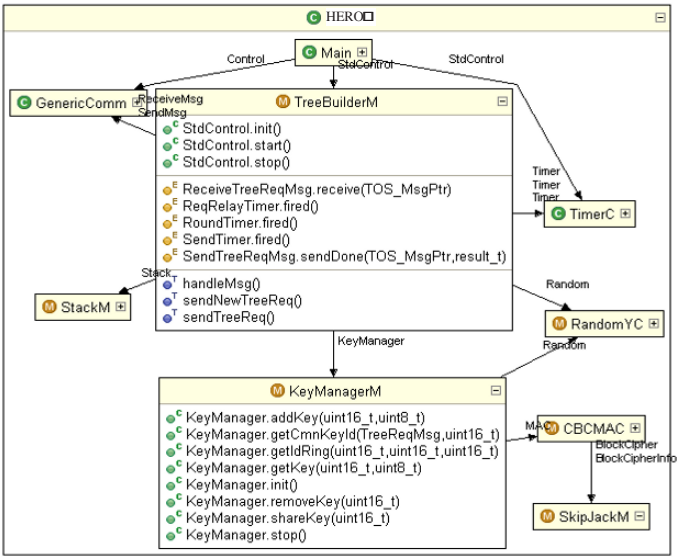
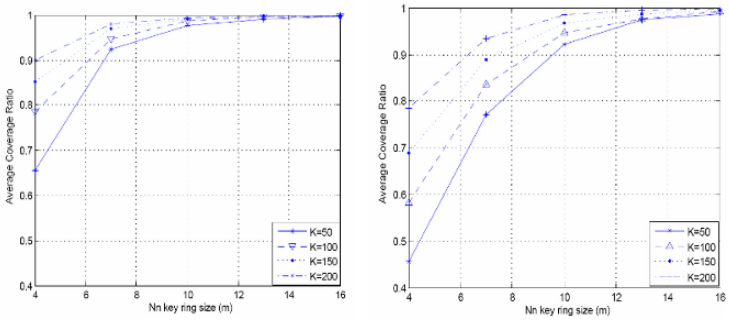


Fig. 4. The components and the interfaces of TinyOS used to implement HERO

5.1 Secure coverage ratio vs. key ring sizes

In this simulation scenario, we considered a 1000 motes network. Mote 0 being the Sink, and 10 motes are considered as Super nodes. We considered different key ring sizes for two cases: a pool of 1000 keys and a pool of 2000 keys. Figure 5 illustrates the evolution of the secure coverage of the network for a 1000 and 2000 key pool respectively. We notice that in both cases the secure coverage of the network increases when increasing the key ring sizes. What is interesting to notice is the fact that with small key rings we can reach very high secure coverage of the network. Namely for a 1000 key pool, we can reach more than 90% of secure coverage with only 7 keys at Nn nodes and 50 keys at Sn. Depending on the resources available at Super nodes, we can reach the same coverage ratio while further decreasing Nn ring size: we can use 6 keys in Nn ring while using 100 keys in Sn ring, or 5 keys in Nn ring while using 150 keys in Sn ring, or only 4 keys in Nn ring while using 200 keys in Sn ring. We also notice that the smaller is the key pool, the greater is the coverage ratio. However, the rekeying overhead is smaller when the key pool is greater. Indeed, with a large key pool only few nodes share the same keys of a compromised node (cf. 3.1). Table 5.1 compares HERO to some solutions of the literature with respect to the probability of key sharing. We notice that HERO induces the smallest key ring storage overhead while achieving the highest key sharing probability.



(a) P=1000 (b) P=2000

Fig. 5. Evolution of secure coverage ratio with respect to key rings sizes

Table 1. Comparison of some key pre-distribution solutions with respect to key sharing probability for Key pool size P=1000

WSN type	Protocol	Parcel key pool size(K)	Key ring size(m)	Probability of sharing a key
Homogeneous	Eschenauer and Gligor's basic scheme	100	100	0.5
Heterogeneous	Asymmetric key pre-distribution scheme (AP)	500	20	0.5
	Brown et al. public-key based scheme	104	43	0.5
Heterogeneous	**HERO**	**100**	**3**	**0.7**

5.2 Secure coverage ratio vs. network size

In this simulation scenario, we considered a key pool size of 1000 keys. Each Super Node is initiated with a ring of 100 keys, and each Normal Node is initiated with a ring of 5 keys. The key rings are randomly chosen from the 1000 key pool. Then we varied the network size from 100 nodes to 1000 nodes. For each point, we run our simulation 10 times and we calculated the average secure coverage ratio of the network. Figure 6 illustrates the evolution of secure coverage ratio when we increase the network size. We notice that the secure coverage ratio depends on the density of the network: the greater is the network density, the higher is the secure coverage ratio. Moreover, for network size in $[100 - 300]$ nodes the secure coverage ratio increases with almost 100%. With only 550 motes, HERO allows to reach more than 90% of secure coverage of the network. This means that with only 1 mote per $450m^2$ we reach more than 90% of secure coverage using HERO.

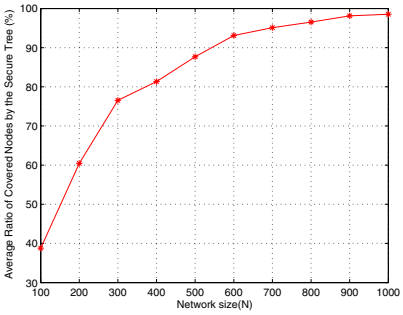


Fig. 6. Evolution of secure coverage ratio with respect to network size

6 Conclusions

In this paper, we have studied random key pre-distribution for wireless sensor networks. This category of key management schemes aims to secure links between sensors of a WSN. Many studies and applications demonstrate that communications in WSN follow mostly many-to-one paradigm. Also, the new works confirm that, the usage of heterogeneous nodes gives better performance than homogenous ones in terms of energy consumptions, storage overhead, and network connectivity. Therefore, we exploited the heterogeneity features, in this paper, to propose a new key management scheme that secures this communication paradigm through constructing a secure tree rooted at the Sink. Our solution (HERO) is designed for heterogeneous sensor networks and relied on random key pre-distribution. HERO secures child-parent links through pairwise shared keys. Preliminary simulation results using TinyOS show that our solution reduces the key storage overhead compared to existing schemes.

References

1. Levis, P., Lee, N., Welsh, M., Culler, D. : TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Application. In: ACM SenSys. Los Angeles(2003)
2. Eschenauer, L., Gligor, V.D. : A key management scheme for distributed sensor networks. In: Proceedings of (CCS '02.), pp. 41-47. Washington(2002)
3. Hussain, S., Kausar, F., Masood, A.: An efficient key distribution scheme for heterogeneous sensor networks. In: the international conference on wireless communications and mobile computing, pp. 388-392.Hawaii(2007)
4. Brown, J., Du, X., Nygrad, K. : An Efficient Public-Key-based Heterogeneous Sensor Network Key Distribution Scheme. In: IEEE GLOBECOM. Washington(2007)
5. Jason Lester H. : System Architecture for Wireless Sensor Network. In:PhD Dissertation, University of California, Berkeley (2003)
6. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K. : System Architecture Directions for Networked Sensors. In: Proceeding of ACM ASPLOS IX. (2000)
7. Xu, N. : A Survey of Sensor Network Applications. In: Survey Paper for CS694a. Computer Science Department, University of Southern California (2002)

8. Camtepe, S. A., Yener, B. : Key Distribution Mechanisms for Wireless Sensor Networks: a Survey. In: TechReport TR-05-07. System Design Laboratory (2005)
9. Yarvis, M., Kushalnagar, N., Singh, H., Rangarajan, A., Liu, Y., Singh, S. : Exploiting Heterogeneity in Sensor Networks. In: IEEE INFOCOM'05. Miami(2005)
10. Gay, D., Levis, P., Behren, R., Welsh, M., Brewer, E., Culler, D. : The nesC language: A Holistic Approach to Networked Embedded System.In: Precedings of (PLDI).California (2003)
11. Du, X., Xiao, Y., Guizani, M.,Chen,H-H.: An effective key management scheme for heterogenous sensor networks. Ad Hoc Networks, Elsevier. **5(1)**, 24–34 ,Jan (2007)
12. Traynor, P., Kumar, K., Choi, H. : Effcient Hybrid Security Mechanisms for Heterogenous Sensor Networks. IEEE Trans.(MTC). **6(6)**, 663–677,Jun (2007)
13. Technology. Crossbow : http://www.xbow.com
14. UC. Berkeley : http://nescc.sourceforge.net