R. Roman, J. Lopez, C. Alcaraz, and H. Hwa Chen, "SenseKey - Simplifying the Selection of Key Management Schemes for Sensor Networks", *5th International Symposium on Security and Multimodality in Pervasive Environments (SMPE11)*, 2011.
http://doi.org/10.1109/WAINA.2011.78
NICS Lab. Publications: https://www.nics.uma.es/publications

SenseKey - Simplifying the Selection of Key Management Schemes for Sensor Networks

Rodrigo Roman, Javier Lopez, Cristina Alcaraz
Computer Science Department
University of Malaga
Malaga, Spain
Email: {roman,jlm,alcaraz}@lcc.uma.es

Hsiao-Hwa Chen
Department of Engineering Science
National Cheng Kung University
Tainan City, Taiwan
Email: hshwchen@ieee.org

***Abstract*—Key Management Schemes (KMS) are a very important security mechanism for Wireless Sensor Networks (WSN), as they are used to manage the credentials (i.e. secret keys) that are needed by the security primitives. There is a large number of available KMS protocols in the literature, but it is not clear what should network designers do to choose the most suitable protocol for the needs of their applications. In this paper, we consider that given a certain set of application requirements, the network designer can check which properties comply with those requirements and select the KMS protocols that contains those particular properties. Therefore, we study the relationship between requirements and properties, and we provide a web tool, the SenseKey tool, that can be used to automatically obtain an optimal set of KMS protocols.**

***Keywords*-Key Management; Security; Sensor Network;**

I. INTRODUCTION

Wireless sensor networks (WSN) can be considered as a sort of virtual skin, as they are able to provide information from the physical world (e.g. humidity, temperature, radiation) to any computer system. The core hardware elements of a WSN, the sensor nodes, are small computers with limited capabilities that are able to "feel" (sense their surroundings), "think" (make use of their computational capabilities), "talk" (communicate with each other using wireless transceivers), and "subsist" (power themselves using batteries or other energy sources). The information retrieved by these sensor nodes is collected by interface devices known as base stations, and provided to any user interested in the information, human or machine.

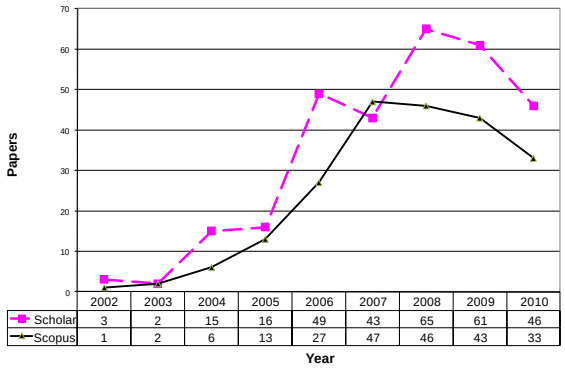
Research on WSN has been very active these last years. In fact, not only there are already some standards that can provide support for WSN applications in home or industrial environments (e.g. ZigBee [4], WirelessHART [5]), but also WSN are seen as one of the foundational blocks for the future Internet of Things [6]. Still, there are some challenges that must be considered on the deployment of any WSN, and one of those challenges is Security. In fact, due to its importance for the bootstrapping of a secure link-layer channel (i.e., based on pairwise key) amongst neighboring nodes, one of the most important security mechanisms that must be implemented in all applications is the Key Management Scheme (KMS). However, many KMS protocols have been proposed in the literature. As they provide different features or properties, it is not an easy task for a network designer to pinpoint the right scheme or protocol for WSN applications without resorting to an exhaustive search.

In this paper, we consider that the requirements of a WSN application are tightly linked with the properties offered by KMS protocols. As a result, we provide an enumeration of the different KMS properties and their relationship with the application requirements. We also use such information as one of the cornerstones of a web-based tool called SenseKey, whose main purpose is to allow network designers to select an optimal KMS protocol in a simple way.

The rest of the paper is organized as follows. In Section II we further explain the importance of the KMS protocols, highlighting the research and industrial efforts to obtain a set of useful protocols. In section III, we enumerate the most important KMS protocol properties, describing how they are linked with the application requirements. Section IV introduces the SenseKey tool, explaining both its underlying algorithms and the design and usage of the web tool itself. Finally, Section V concludes the paper.

II. KMS PROTOCOLS

Every system must be properly secured against malicious threats that can affect its functionality, and WSNs are not an exception. In fact, WSNs are especially vulnerable against external and internal attacks due to their peculiar characteristics. The devices of the network are usually constrained, thus it is challenging to implement the necessary security protocols. It is usually easy to physically access the sensor nodes, allowing any experienced attacker to extract information from the node. Finally, the wireless communication channel can be easily accessed and attacked, and in fact due to its distributed nature a WSN can be attacked at any point. Therefore, it is necessary to develop lightweight security mechanisms which can provide a basic layer of protection for WSN applications.



	2002	2003	2004	2005	2006	2007	2008	2009	2010
Scholar	3	2	15	16	49	43	65	61	46
Scopus	1	2	6	13	27	47	46	43	33

Figure 1. KMS Protocols Evolution.

One of the most important security mechanisms that must be implemented in any WSN deployment is the Key Management Scheme, or KMS. As most sensor networks communicate through a wireless channel with other nodes, it is necessary to protect this channel from attacks such as message injection, eavesdropping, and packet relaying. This protection is mainly provided by the security primitives, such as symmetric key cryptography (SKC) and public key cryptography (PKC). However, these primitives need of certain security credentials (e.g. secret keys shared between nodes, public key certificates) in order to be able to open a secure channel. Precisely, the main tasks of a KMS protocol is to create, distribute, and maintain such security credentials.

Due to its importance as one of the foundations for the development of secure WSNs, the design of KMS protocols has been one of the priorities of the research community these last years. One example of this claim can be found in Figure 1, which shows the number of unfiltered articles whose title contains the words "Key Management" and "Sensor Networks" in both Google Scholar and Scopus [1]. From the figure we can deduce that the number of KMS protocols presented every year has been constantly growing, although as of 2010 we already reached a peak point. Another example of the previous claim is the high number of KMS surveys that have been produced by the research community (cf. [2], [3]). These surveys have studied the different types of KMS protocols that are available in the literature, providing in certain cases a detailed overview of some KMS properties and features.

In fact, by analyzing the previous surveys, it is possible to classify most KMS protocols into four categories. The *"Key Pool" Framework*, whose first protocol was introduced by Eschenauer and Gligor [9], considers that a node stores a small subset of keys retrieved from a global key pool. If two nodes find a common key, they can use them to establish a session key. In the *Mathematical Framework*, certain KMS protocols (such as [10]–[12]) use mathematical concepts (Linear Algebra, Combinatorics, and Algebraic Geometry) for calculating the pairwise keys of the nodes. The *Negotiation Framework* has a more simplistic approach, where nodes negotiate all shared keys after the deployment (cf. [13], [14], [18]). Finally, the *Public Key Framework* relies on PKC to securely bootstrap the pairwise key of two nodes over a public communication channel.

All the previous frameworks make use of somewhat complex approaches and different strategies to provide desirable properties such as network resilience (i.e. the ability to cope with stolen credentials and rogue nodes). However, existing WSN standards such as ZigBee [4] and WirelessHART [5] use simple negotiation-based KMS protocols. ZigBee PRO makes use of a network-wide 'network key', which protects all communications at the network level and is shared by all the nodes of the network, and a 'link key', which is used to protect the communication channel between the node and the central server. As for the key deployment mechanism, a key known as 'master key', preconfigured in all devices, is used to generate the link key by means of a Symmetric-Key-Key-Exchange (SKKE) algorithm. Afterwards, the network key is transmitted encrypted with the link key. As for WirelessHART, it manages four different types of security keys. There are two preconfigured keys, 'public key' and 'join key', which are used to bootstrap a link-layer pairwise key shared between two nodes ('session key') and a key shared by all network devices ('network key').

By using simple KMS protocols, these standards can provide a general solution that fulfills the requirements of many applications while consuming as less resources as possible. Still, there are other scenarios that need of different KMS protocols with specific properties in order to fulfill their particular requirements. For example, one property that has been widely considered in all KMS surveys is *network resilience*. If we have a very critical environment where a subverted node must cause as less damage as possible to the network, the use of network-wide keys that can be used by anyone to obtain the global session key of the network should be discouraged. Another example is the *communication overhead* property: some environments (e.g. Underwater Sensor Networks) should send as less messages as possible in order to bootstrap the security credentials due to diverse factors such as energy consumption, channel constraints [8], security concerns, and others.

III. MAPPING APPLICATION REQUIREMENTS TO KMS PROPERTIES

It has been shown in the previous section that there are many KMS protocols, either created by the industry or developed by the academia, that can be used to distribute the security credentials. One problem that network designers

[1] http://scholar.google.com/,http://www.scopus.com/

face for the creation of a particular WSN application is to choose the most suitable KMS protocol for their needs. In fact, some protocols may seem to be better than others, but it is not exactly clear which one performs the best. One particular approach that can be effectively used to solve this problem is to analyze the relationship between the application requirements and the properties of the KMS protocols: Given a certain set of application requirements (e.g. the network contains thousands of nodes), the network designer can check which properties comply with those requirements (scalability), and select the KMS protocols that contains those particular properties.

In order to achieve this particular approach, it is necessary to complete two tasks: i) discover and enumerate the most important KMS protocol properties, and ii) describe how they are linked with the application requirements (e.g. the *Scalability* property is not important in WSNs whose size is small). A summary of the results from both tasks is presented in the following paragraphs. For obtaining these results, we have studied the security and operational requirements of sensor networks relevant to the key management, and we have also analyzed the actual state of the art and the existing surveys in the area.

1) Memory Footprint (Mm): A sensor node is usually constrained in terms of memory (e.g., < 16 kB of data memory and < 256 kB of instruction memory). Therefore, it is important to know what is the amount of data memory that a KMS protocol requires for storing the security credentials used for bootstrapping the entire infrastructure. The amount of instruction memory that a KMS needs to implement the whole protocol is also important.

The importance of this property is closely associated with the complexity of a particular application. In general, the simpler the application is, the more space can be used for storing security credentials.

2) Communication Overhead (Cm): In most KMS protocols, the nodes must exchange information with their peers through communication channels in order to establish their pairwise keys. While some protocols may require the exchange of a small amount of information, the other protocols may need to undergo complex negotiation processes among peers.

There are many WSN scenarios where the communication overhead should be reduced to a minimal level. For instance, when an application needs to establish the links to provide its services within a very short period of time.

3) Processing Speed (Sp): Sensor nodes are usually severely constrained in terms of their computing power. Fortunately many KMS protocols are not very computationally intensive. As for the communication overhead property, the time consumed in negotiating the pairwise keys mainly includes the durations in sending and receiving messages through the wireless channels. Nevertheless, some protocols may indeed require some computational-intensive tasks.

The processing times for different key management schemes are associated closely with the communication overhead required by the KMS. In particular, some WSN applications (e.g. mobiles nodes moving at a fast speed) may require to set up a secure channel between two previously unknown nodes as fast as possible. Thus, a fast establishment of the communication link is critical, and a reduction in the overhead can facilitate to shorten the link establishment time.

4) Network Bootstrapping (Sec): While the whole key distribution process must be secure by itself, some protocols do assume that the network is less likely to be in danger right after its deployment, and choose to exchange some secret information without any protection whatsoever. Other protocols provide partial (and usually harmless) information about the secret key (e.g. the index of the keys from a key chain), and finally some protocols do not exchange sensitive information at all (e.g. exchanging their IDs).

For the applications where the deployment environment is secure enough, confidentiality in the bootstrapping process is not an issue since there are no attackers in the area that may steal the security credentials. However, the problems may arise whenever the deployment area is open to public or when the information managed by the sensor nodes is important.

5) Network Resilience (Rs): Network resilience indicates the ability to cope with stolen credentials and rogue nodes. The higher the network resilience is, the lower the chance is for a malicious attacker to control a significant part of the network using the stolen credentials. The best resilience is offered by the KMS protocols where nodes share pairwise keys only with their direct neighborhood.

If the environment where a sensor network is deployed is heavily protected or the probability of capturing a set of nodes is extremely low, then the network resilience is not a critical factor. Consequently, the need for network resilience increases with the chance of a node being subverted by an adversary.

6) Connectivity: Connectivity is a property which is related to the capability for two sensor nodes to share the same security credentials. There are three main connectivity properties, as listed below.

- Global connectivity (GC) is the ratio between the largest size of isolated components in the network and the size of the whole network. If the GC is 100%, it means that there is always a key path, i.e., a secure routing path, between any two nodes in the network.
- Local connectivity (LC) can be defined as the probability that two neighboring nodes share the same secret key right after the network starts to operate. If LC is 100%, then any node can securely communicate with any of its neighbors without negotiation.
- Node connectivity (NC) is the probability of any two nodes of the network to share one secret key, regardless of their location in the network. If NC is 100%, then

any node in the network can open a pairwise secure channel with any other node.

Global connectivity is usually quite important, as in many scenarios all nodes are equally important for providing the network services. Local connectivity is also important, as in most cases the locations of the nodes are unknown and they must be able to create a pairwise key with their neighbours. Finally, node connectivity is indispensable in some WSN application scenarios where nodes are mobile, requiring to open a secure channel with any node in their vicinity.

7) Scalability (Sc) and Extensibility (Ex): A KMS protocol offers the scalability if it is able to support a WSN network with a large number of nodes (at least at an order of thousands). On the other hand, a KMS protocol is extensible if it allows the inclusion of new nodes after its initial deployment.

Scalability is not an important issue for the WSN applications that require a small number of sensor nodes (i.e., less than 100). However, as the size of the network increases, the scalability becomes more important. The extensibility property is important for those scenarios where there may exist hostile external entities. It is also important for those applications which have to provide a service for a relatively long period of time (where there should exist some maintenance policies).

8) Energy (En): A sensor node usually relies on batteries for powering itself. Since the establishment of pairwise security credentials between nodes is an energy-consuming task (transmitting the security credentials, sending/receiving data to/from other peers, etc.), it is important to consider how much energy is consumed during the operation of a KMS protocol.

On the other hand, there are also some scenarios where energy saving is not a critical factor. For instance, in WSNs with a relatively short lifetime or in network configurations whose nodes can access to unlimited energy sources, such as solar energy or even normal power lines.

IV. SENSEKEY: CHOOSING A KMS PROTOCOL

In the previous section, we have discussed that network designers can choose a KMS protocol that suits the needs of their WSN applications by mapping the requirements of such applications to the KMS properties. However, this is actually a cumbersome task if performed manually: network designers must figure out which are the specific properties that are provided by the KMS protocols (although some of the surveys can be used to complete this task, such as [3]) and study the suitability of every KMS protocol given the application requirements.

As network designers need of an automated tool that can provide them with the information they need, we have designed a web tool, SenseKey. The core of SenseKey is a well-defined algorithm that uses as an input i) the properties that are essential for a KMS in the context of the WSN application (*main properties*), and ii) the properties that are important but not essential (*secondary properties*). The algorithm uses this input to apply a set of filters to a list of KMS protocols, and provides as an output the most suitable protocols. Note that a partial version of this algorithm has been defined before [7], but neither was complete (i.e. it was not possible to automate its behaviour) nor it considered all the KMS properties presented in the previous section.

A. KMS SELECTION ALGORITHM

The first step of the algorithm (step 0) consists of constructing the table shown in Table I, which provides the information on how different protocols fit to the properties introduced in Section III. Due to space restrictions, the table includes only some of the most relevant KMS protocols in the literature. Note that this table can be easily extended, just by inserting an additional row with the properties of a new protocol. Some properties are labelled as design-dependant, since there are protocols that can be tweaked in order to offer better properties. Also, it is indicated whether a certain protocol requires particular deployment knowledge.

Based on the table generated in step 0, the following series of well-defined steps take place:

1) The network designer identifies the properties of the specific WSN scenario or application by analyzing its requirements, according to the descriptions given in Section III. Then, he/she decides which of those properties are essential for a KMS in the context of the WSN application (*main properties*), and which are important but not essential (*secondary properties*).
2) The network designer selects from Table I all the KMS protocols that fulfill one or more of the *main properties* with the sign "++" or "+". These protocols conform the *KMS candidate set* for the particular WSN application.
3) From the KMS candidate set, the network designer discards all the protocols that have one of its main properties or secondary properties with the sign "−−" or "−". Note that if a property is affected by the variables used in the design of the protocol, it should be ignored. As a result, the KMS candidate set will only contain the protocols that are at least good enough to fit the major properties of the application.
4) From the resultant KMS candidate set, the network designer discards all the protocols that do not have every main properties marked with the sign "++" or "+". As a result of this step, the KMS candidate set will contain the protocols that provide the essential properties required for the WSN application.
5) Finally, the network designer reviews the protocols still contained in the KMS candidate set and choose the most suitable one for the application. Note that it is possible to order the protocols inside KMS candidate set in terms of their properties. Also, the network

Table I
MAJOR PROPERTIES FOR WIDELY USED KMS

Protocol	Cm	GC	LC	NC	En	Ex	Mm	Rs	Sc	Sec	Sp
Key Infection [13]	−	++	++	−−	+	−−	++	++	++	−−	+
SAKE [14]		++	++	−−		++		++	++		−−
Generalized Quadrangle [11]	−−	++		−−		−−	++				++
Symmetric Design [11]	++	++	++	++		−−	++	−−			++
Hybrid Designs - Generalized Quadrangle [11]		○	○	○		−	++		○		++
Basic Probabilistic Key Pre-distribution [9]	−	−	−	−			○	−	++		
Random Key Pre-distribution [20]	+	+	+	−−			○	−	++		
Blom Key Pre-distribution [10]	++	++	++	++		−−	○	○	○	+	
Multiple Space Key Pre-distribution [10]	−	−	−	−	−−		○	++	++		
Multiple ID-Based one-way Function [15]	−−	++	++	−−		−−	++	−−	++		++
Multiple Space Blom (MBS) [15]	++					−−	○	○	○		
Deterministic Multiple Space Blom DMBS [15]	−	−	−	−			○	++	++		
Robust Continuity Blom [21]	+	++	++	++		++	○	○	+	+	−
Grid Based Key Pre-distribution [12]	++	++	++		−−	−−	○	++	○		−−
Polynomial Based Key Pre-distribution [12]	++	++	++	++	−−	++	○	○	○	++	−−
Random Subset Assignment [12]	−	−	−	−	−−		○	○	++		−−
PIKE [16]	−	−	−	−		−	++		++		
LEAP+ [17]		++	++	−−		++		○	++		++
Panja [18]		++	++	−−		−−	++	−−	++		−−
RPB Scheme [19]	++	++	++	++		++	−	+	++	++	
Public Key Cryptography-based KMS [22]	++	++	++	++	−−	++		++	++	+	−−

Notation

Advantage	++
Disadvantage	−−
Advantage, depending on the design of the protocol	+
Disadvantage, depending on the design of the protocol	−
Either advantage or disadvantage, depending on the design of the protocol	○
Node locations are known before deployment	

designer must pay attention to the design parameters of a protocol if any of the main properties or secondary properties are dependant on them.

It must be noted that the algorithm may produce an empty KMS candidate set, meaning that there is no protocol that can completely fulfill the application requirements. However, it is important to realize that it is still possible to manually identify the protocols that partially comply with steps 3 and 4 in the method. Therefore, it is possible to apply one of those partially suitable protocols for that scenario.

B. SENSEKEY: THE WEB TOOL

As a second step in our goal to facilitate the selection of a set of KMS protocols, we have implemented the core of SenseKey as a web tool[2]. In this tool, the user has to enter the main properties and secondary properties of the WSN application, together with other information (e.g., the existence of deployment knowledge, etc.). Once the properties are selected, the SenseKey tool will output a list of suitable KMS protocols. For every selected protocol, the tool provides a list of its advantages (i.e. properties fulfilled by the protocol) and disadvantages (i.e. properties not fulfilled by the protocol), pointing out the properties that are considered design-dependant and highlighting the properties that were required by the network designer. The user of the tool can also check anytime the scholarly article that explains the protocol by clicking its name.

Internally, Sensekey is designed with the purpose of separating the logic of the application from the presentation of the information. Whenever the user requires the application to provide a set of suitable KMS protocols, a controller class initializes the model (retrieving the database of protocols from a XML file) and executes the Sensekey algorithm. After that, a viewer class provides an user-tailored output (using a printer-friendly css template or a web-oriented one), generating the user interface by querying the actual state of the model. The web application is implemented using JavaServer Pages over tomcat.

Finally, as this tool is meant to be used by a wide range of users, both WSN experts and non-experts alike, we have followed certain usability principles in order to make an user-friendly tool. The tool includes a manual that explains not only how the tool itself should be used, but also how to find which properties can be considered main properties or secondary properties. Besides, the interface of the tool also offers context-sensitive help for every listed property. Moreover, the output of the tool show the protocols in order of relevance, that is, the protocols who comply with more properties are shown first.

[2]http://www.lcc.uma.es/~roman/KMSCRISIS

V. CONCLUSIONS

In order to help network designers to decide on a particular KMS protocol, this paper has described how the properties characterizing a protocol are related to the requirements of WSN applications. This leads to our proposed KMS selection algorithm, which is the cornerstone of the SenseKey web tool. The results provided by this paper can be used as a foundation for a) creating a similar tool for cluster-based KMS, and b) analyzing the KMS properties that need to be fulfilled by the current state of the art.

VI. ACKNOWLEDGMENTS

This work has been partially supported by the ARES project (CSD2007-00004) and the SPRINT (TIN2009-09237) project, being the last one also co-funded by FEDER. The third author has been funded by the Spanish FPI (Formacion de Personal Investigador) Research Programme.

REFERENCES

[1] Y. Zhou, Y. Fang, Y. Zhang. *Securing Wireless Sensor Networks: a Survey*. IEEE Communications Surveys & Tutorials, Vol. 10, No. 3, pp. 6-28, 2008.

[2] Y. Xiao, V. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, *A Survey of Key Management Schemes in Wireless Sensor Networks*. Computer Communications, Vol. 30, No. 11-12, pp. 3214-2341, 2007.

[3] M. A. Simplicio Jr, P. S. L. M. Barreto, C. B. Margia, T. C. M. B. Carvalho. *A Survey on Key Management Mechanisms for Distributed Wireless Sensor Networks*. Computer Networks, In Press.

[4] ZigBee Alliance, http://www.zigbee.org/, Accessed on November 2010.

[5] HART Communication Foundation, http://www.hartcomm.org/, Accessed on November 2010.

[6] V. Ovidiu, M. Harrison, H. Vogt, K. Kalaboukas, M. Tomasella, K. Wouters, S. Gusmeroli, S. Haller, Internet of Things Strategic Research Roadmap, European Commission - Information Society and Media DG, 2009.

[7] C. Alcaraz, R. Roman. *Applying Key Infrastructures for Sensor Networks in CIP/CIIP Scenarios*. Proceedings of the 1st International Workshop on Critical Information Infrastructures Security (CRITIS 2006), pp. 166-178, Samos (Greece), 2006.

[8] L. Liu, S. Zhou, J. H. Cui. *Prospects and Problems of Wireless Communications for Underwater Sensor Networks*. Wireless Communications and Mobile Computing - Special Issue on Underwater Sensor Networks, Vol. 8, No. 8, pp. 977994, 2008.

[9] L. Eschenauer, V.D. Gligor. *A Key-management Scheme for Distributed Sensor Networks*. Proceedings of the 9th ACM conference on Computer and communications security (CCS '02), pp. 41-47, 2002.

[10] W. Du, J. Deng, Y. S. Han, P. Varshney, J. Katz, A. Khalili. *A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks*. ACM Transactions on Information and System Security (TISSEC), Vol. 8, No. 2, pp. 228-258, 2005.

[11] S. A. Camtepe, B. Yener. *Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks*. IEEE/ACM Transactions on Networking, Vol. 15, No. 2, pp. 346-358, 2007.

[12] D. Liu, P. Ning, R. Li. *Establishing Pairwise Keys in Distributed Sensor Networks*. ACM Transactions on Information and System Security, Vol. 8, No. 1, pp. 41-77, 2005.

[13] R. J. Anderson, H. Chan, A. Perrig. *Key Infection: Smart Trust for Smart Dust*. Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP 2004), pp. 206-215, 2004.

[14] A. Seshadri, M. Luk, A. Perrig. *SAKE: Software Attestation for Key Establishment in Sensor Networks*. Proceedings of the 2008 International Conference on Distributed Computing in Sensor Systems (DCOSS'08), pp. 372-385, 2008.

[15] J. Lee, and D. R. Stinson. *On the Construction of Practical Key Predistribution Schemes for Distributed Sensor Networks using Combinatorial Designs*. ACM Transactions on Information and System Security (TISSEC), Vol. 11, No. 2, 2008.

[16] H. Chan, and A. Perrig. *PIKE: Peer Intermediaries for Key Establishment in Sensor Networks*. Proceedings of the 24th Conference of the IEEE Communications Society (Infocom05), Vol. 1, pp. 524-535, 2005.

[17] S. Zhu, S. Setia, and S. Jajodia. *LEAP+: Efficient Security Mechanisms for Large-scale Distributed Sensor Networks*. ACM Transactions on Sensor Networks, Vol. 2, No. 4, pp. 500-528, 2006.

[18] B. Panja, S. Madria, and B. Bhargava. *Energy and Communication Efficient Group Key Management Protocol for Hierarchical Sensor Networks*. Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06), pp. 384-393, 2006.

[19] W. Zhang, M. Tran, S. Zhu, G. Cao. *A Random Perturbation-Based Scheme for Pairwise Key Establishment in Sensor Networks*. Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing (MOBIHOC 2007), pp. 90-99, 2007.

[20] J. Jaworski, M. Ren, K. Rybarczyk. *Random Key Predistribution for Wireless Sensor Networks Using Deployment Knowledge*. Computing, Vol. 85, No. 1-2, pp. 57-76, 2009.

[21] M. Wen, Y.-F. Zheng, W.-J. Ye, K.-F. Chen, W.-D. Qiu. *A Key Management Protocol with Robust Continuity for Sensor Networks*. Computer Standards and Interfaces, Vol. 31, No. 4, pp. 642-647, 2009.

[22] A. Liu, P. Ning, *TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks*. Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), SPOTS Track, pp. 245-256, 2008.