SKEW: An Efficient Self Key Establishment Protocol for Wireless Sensor Networks

Mohsen Sharifi, Saeid Pourroostaei Ardakani, Saeed Sedighian Kashi
Computer Engineering Department, Iran University of Science and Technology
msharifi@iust.ac.ir, spourroostaei@gmail.com, sedighian@iust.ac.ir

ABSTRACT

Since wireless sensor networks continue to grow in usage and many sensor-based systems reside in adversarial environments, security consideration is really vital for these systems. But one of the main challenges for the efficient distribution of security keys in wireless sensor networks is the resource scarcity. This paper presents an efficient Self Key Establishment protocol for Wireless sensor networks, nicknamed SKEW, in support of in-network processing. We show that SKEW manages keys with less storage, communication, key transmission frequency, and computational overheads in comparison with similar protocols for the same purpose. All of these benefits are attained by usage of a very few number of messages for key distribution. Since SKEW preserves the network security even before start up time, it can well serve as a base security protocol for all types of security protocols in wireless sensor networks. In this protocol, none of the sensors in the network can send any packets without encryption. It also uses a key refreshing mechanism that prolongs the network security. Smart dust networks and pervasive computing environments can particularly benefit from the proposed protocol.

KEYWORDS: Wireless Sensor Networks, Security Protocol, Key Distribution, Key Establishment, Clustering, Distributed Wireless Sensor Networks.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) are quickly gaining popularity due to the fact that they are potentially low cost solutions to a variety of real-world challenges [1].
WSNs architectures can generally be organized in two ways: distributed and hierarchal as shown in Figure 1 [11]. A hierarchical WSN has a network hierarchy among the sensor nodes based on their properties such as power and memory.

Cluster heads which are used to collect and aggregate local or received data from other end-point sensor nodes and send them to base stations [11]. Data communications in such networks can be: (1) pair-wise (unicast), (2) group-wise (multicast) or (3) network-wise (broadcast).

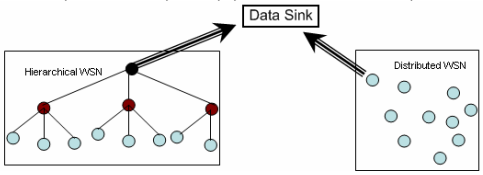


Figure1: Distributed versus hierarchical WSNs.

There are some lightweight key management protocols [2, 4, 5, 7, 8] for WSNs. Cryptography keys in these protocols are transmitted within nodes via messages. So they incur high communication overheads [14, 15]. To see why, let's consider two communicating nodes in a secure session. A sender node must either send 2 messages to a receiver node, one for transmitting its symmetric key and another for the message text itself, or just send 1 message containing the text of its message if it knows (has stored) the symmetric keys of all its neighbors.

Distributed and hierarchical WSNs architectures [11] require different key distribution protocols. In distributed WSNs architectures, sensor nodes use either pre-distributed, dynamically generated pair-wise or group-wise keys [12, 13, 16, 21]. Any key distribution mechanism must be fit and efficient for the type of key usages.

In hierarchical WSNs, there are some trusted nodes, such as base station or cluster heads, which act as the key server. Trustees distribute keys [22, 23, 24, 25] by a secure session establishment.

In this paper, both hierarchical and distributed architectures are considered. Firstly, a hierarchical architecture, with a single base station and many clusters, is considered. The base station is the network coordinator with which cluster heads in its radio range can communicate. Since sensor nodes have limited radio

coverage, the network is clustered in such a way that each cluster head can communicate to the base station in a single hop; ordinary (end-point) nodes within a cluster communicate with their cluster head in a single hop too. Secondly, a distributed architecture without a pre defined clustering is considered. In this case we have a single base station and many sensor nodes, so each node can communicate with the base station in a single hop. A reconfigurable clustering for key distribution is introduced for this architecture too [17, 18, 19].

The rest of paper is organized as follows. Section 2 describes the three most relevant security protocols for WSNs, namely SPINS [2], SNAKE [6], BROSK [6] and LEAP [8] key management protocols. It describes other similar extensions to these protocols too. Section 3 presents the SKEW protocol and the assumptions on which it is based. Section 4 discusses the performance evaluation of SKEW, and the last section concludes the paper and presents some future thoughts.

2. RELATED WORK

The key management protocols for WSNs most relevant to SKEW are SPINS, SNAKE, BROSK and LEAP protocols.

2.1 SPINS

SPINS (Security Protocols for Sensor Networks) is a security protocol that includes two protocols, SNEP, µ-TESLA [3]. SNEP provides data confidentiality, two-party data authentication and data freshness, and µ-TESLA provides authenticated broadcast for severely resource-constrained environments.

In this protocol, the base station (Key Server) assigns a unique key to each session for communication between any pair of nodes.

All cryptographic primitives, i.e. encryption, message authentication code (MAC), hash, and random number generator, are constructed out of a single block cipher for code reuse. This, along with the symmetric cryptographic primitives used reduces the overhead on the resource constrained sensor network.

In a broadcast medium such as a sensor network, data authentication through a symmetric mechanism cannot be applied as all the receivers know the key. µ-TESLA constructs authenticated broadcast from symmetric primitives, but introduces asymmetry with delayed key disclosure and one-way function key chains [6].

2.2. SNAKE

SNAKE is a protocol that can negotiate the session key in an ad-hoc way. Nodes do not need a key server to perform key management [7]. For example as is shown in Figure 2 [6], node A which wishes to start communication with node B, sends a request message alongside with a nonce number (N_A) to B. B replies with a two part message: T and MACK [T]. T includes the identifier of A (ID_A), the identifier of itself (ID_B), the nonce number taken from A (N_A), a nonce number generated by itself (N_B). N_A and N_B are used for data freshness, and MAC_K [T] acts as a message authentication code for A. When A receives this message from B, it checks the MAC and understands that B is a valid node to communicate with. In order for B to get the validity of A as well, A sends a message back to B containing its identifier (ID_A), the nonce number taken from B (N_B), and an authentication code named MACK [$ID_A|N_B$]. Up to this point, A and B become authenticated to each other. Now a shared session key is generated by both nodes ($K_{AB} = MAC_K [N_A|N_B]$) which can be used in their further communications.

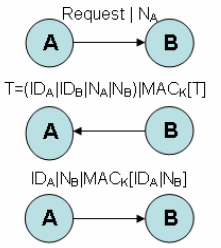


Figure 2: A sample key establishment sequence in SNAKE.

2.3. LEAP

LEAP which stands for Localized Encryption and Authentication Protocol [8] is a key management protocol for sensor networks designed for in-network processing. Every node is only engaged with a limited number of its neighboring nodes to build its required keys out of its neighboring nodes; in other words, it does not involve all nodes of the network. The design of the protocol is motivated by the observation that different types of messages exchanged between sensor nodes have different security requirements, and that a single keying mechanism is not suitable for meeting these different security requirements. Hence, LEAP supports the establishment of four types of keys for each sensor node: an individual key shared with the base station, a pair-wise key shared with another sensor node, a cluster key shared with multiple neighboring nodes, and a group key that is shared by all the nodes in the network.

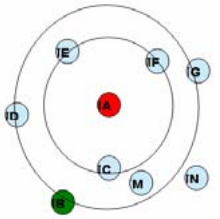
The protocol used for establishing and updating these keys is communication and energy efficient, and

minimizes the involvement of the base station. LEAP also includes an efficient protocol for inter-node traffic authentication based on the use of one-way key chains. A salient feature of the authentication protocol is that it supports source authentication without precluding in-network processing and passive participation [9].

2.4. BROSK

BROSK is another key management protocol that stands for BROadcast Session Key Negotiation Protocol. In this protocol each node can negotiate a session key with its neighbors by message broadcasting. BROSK can be deployed in a large-scale sensor networks and Ad Hoc networks. In this protocol each sensor node, such as A, broadcasts $ID_A|\ N_A||MAC_K\ (ID_A|N_A)$ message to all its neighbors as shown in Figure 3 [6]. Every receiving node responds by broadcasting a reply message; e.g. node B broadcasts the $ID_B|\ N_B||MAC_K\ (ID_B|N_B)$ message. A shared session key can then be generated accordingly; for example, the following session key is generated and established between A and B nodes:

Node B: IDB| NB||MACK (IDB|NB)



Node A: IDA| NA||MACK (IDA|NA)

Figure 3: Message broadcasting in BROSK protocol.

3. SKEW APPROACH

We describe our approach in two cases: hierarchical WSNs and distributed WSNs. In the first case, our network is a hierarchical WSN and each sensor node has: A unique ID, A pseudo-random function [10] (F) for generating the next key in sequence, A unique cluster number for each cluster member, and A group key as shared key between all nodes.

We divide node memory to three logical parts: 1) RAM memory section, 2) executive code memory section, and 3) non volatile memory section. Some of these logical memory sections can be in one physical hardware unit.

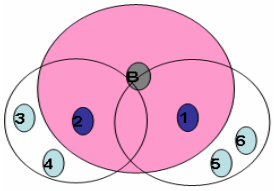
An attacker can steal information which is in executive code and non volatile memory sections but it cannot steal information that is in RAM. If an attacker desires to access RAM information, the node detects this situation and will reset. Hence RAM information will be unavailable. We also suppose that data that is in executive code can be changed.

Each node in the hierarchical approach sends an encrypted message by a cluster key that is generated periodically by the F function in each cluster. This function is invoked like this at special time intervals:

$$K_{vi} \leftarrow F_{vi}\ (K_{vi-1})$$

Each cluster key has a unique version number named V_i. This version in each cluster acts as a sequence number for the next key generation in sequence.

The base station node sends a message containing the initial cluster key (K_{v1}) and the cluster number, encrypted by the group key, to all nodes in each cluster; these keys can be distributed in the network as pre-distributed keys too. Then all receiver nodes (typically cluster heads) forward their messages to all nodes in their cluster. If some node, say H, in a cluster cannot receive the Kv1, it sends a request message encrypted by the group key to its neighboring nodes within the same cluster that have already received the initial cluster key (K_{v1}). This request includes the agreed upon cluster number. Now H can take out the initial cluster key (K_{v1}) from the message of any one of the responding neighbors whose cluster number is the same as H's cluster number.



The Figure 4: Hierarchical WSN approach.

In our approach, every key refreshing message has two parts: a header and trailer as shown in Figure 5. The trailer contains the message text body and the header contains some information about the message such as its cluster key version and cluster number. The header is encrypted by the group key and the trailer is encrypted by a new cluster key.

Key version(Vi)	Cluster number	Message text body encrypted by K_{vi}
Message Header		Message Trailer

Figure 5: Key Refreshing Message format for hierarchical case.

Each node that receives a key refreshing message, it decrypts the message header by the group key and reads the message key version. Then it generates a new cluster key by invoking F and tries to decrypt the message trailer with it. If it can decrypt the tailor successfully, it

continues. Otherwise, it ignores the message on ground of being insecure or tampered with.

The second scenario happens when the network is a distributed WSN with no assumed clustering, wherein each sensor node has: A unique ID, A private key known to the base station too, A pseudo-random function [10] (F) for generating the next key in sequence, A group key as a shared key between all sensor nodes.

As in the hierarchical WSNs, node memory in distributed WSNs is divided to three logical parts: 1) RAM memory section, 2) executive code memory section, and 3) non volatile memory section.

In the distributed WSNs, all nodes encrypt messages with group key and the group key can be refreshing periodically. So each node which generates new version group key, broadcasts the group key to all nodes that can receive key refreshing message.

As shown in figure 6, key refreshing message format is different with the message in hierarchical case because the nodes in distributed case have not cluster number.

Key version(Vi)	Message text body encrypted by K_{vi}
Message Header	Message Trailer

Figure 6: Key Refreshing Message format for distributed case.

In contrast to hierarchical clustered WSNs, wherein self key establishment was restricted to some partitions of the network, so in this paper we describe an approach for clustering. For distributed WSNs transforming to hierarchical WSNs, the base station node sends a hello message (encrypted by the shared group key) to all nodes in its radio range. Each node in the base station's radio range that receives this message starts to get the identifiers of its immediate neighbors. Therefore, the receiver nodes send a message encrypted by group key to base station. This message includes the neighbors' id list. Base station receives these messages and selects the best nodes for coverage as cluster heads. Base station then sends a message (encrypted by group key) containing a unique Kv1 and a cluster number for each cluster head. Every cluster head that receives this message encrypts the message by group key and then broadcasts it to all the nodes in its radio range. Now all nodes have Kv1 and their cluster number. However, some nodes, such as 4 in Figure 7, may reside within the coverage point of more than 1 cluster head. In this case, node 4 selects a cluster number randomly. So at the end the keys are established in the network as in a hierarchical way.

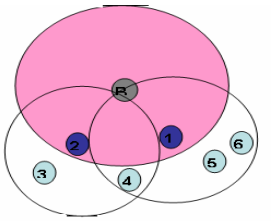


Figure 7: Clustering Approach in SKEW.

Moreover, we can change the group key, pseudo function and other securely information by refreshing message tailor to all nodes periodic. So, probability of those transpire can be lower.

4. PERFORMANCE EVALUATION DISCUSSION

In this paper we select BROSK and LEAP protocols as benchmark for evaluation. As related work section discussions, we found the BROSK protocol has a distributed structure so all nodes distribute randomly on the environment, However in LEAP protocol all nodes distribute on a hierarchically structure.

The following metrics are often used for the performance evaluating of key management scheme [20]:
Connectivity (local/global): local connectivity is the probability of at least one key sharing between two neighbor nodes. The global connectivity is ratio of the numbers of nodes that can earn the new key with communicating to the network size.

Resilience to sensor nodes capture: resilience is the fraction of total keys information exposed to adversary. Scalability: the possibility that new nodes might be added later. Memory efficiency: the amount of memory that used for key storage.

As all nodes in the proposed protocol shared the group key as global key so local connectivity is accepted. For global connectivity proving, we implement our proposed protocol in both distributed and hierarchical cases with VisualSense simulator [29] as shown in figure 9. In first experiment we distributed 16 sensor nodes with 100 meter radio range on 500x500 meter dimensions in 60 second. So key server sensor nodes generate new version key and distributes on the network, each sensor node which received the message, refresh its key version.

In both experiments all sensor nodes can received the message, therefore we increment nodes number to 32, 48, 64 and 100 nodes. Experimental results shown in figure 10 describes global connectivity rate in the networks for our proposed protocol versus LEAP and BROSK protocols. Really we can see connectivity for key

distribution in both cases for SKEW is better than other related protocols.

Actually related protocols have some assumptions for resilience to sensor nodes capture, for example LEAP protocol deleted all securely information such as key generator function [8] and BROSK protocol assumed global key never disclosed [6]. In proposed protocol our assumption about unavailable RAM information can protect from securely information when the nodes captured by adversary.

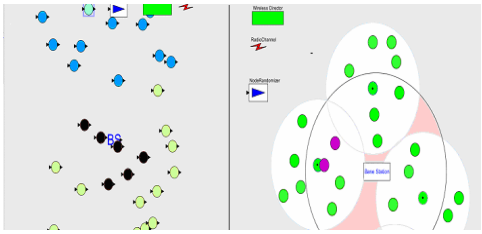


Figure 9: Distributed and Hierarchical Simulation.

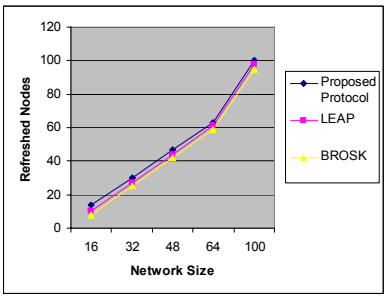


Figure 10: Global connectivity and Scalability.

Growing network size in our experiments proves scalability. We start the simulation experiments in 4 steps and in each step we increment the nodes number to 32, 48, 64 and 100 nodes respectively. So in each step we can see all nodes can use broadcasted instruction to keys refreshment. Experimental results shown in figure 10 describe the missed nodes numbers in the networks for our proposed protocol versus LEAP and BROSK protocol. Really we can see scalability for key distribution in both cases for SKEW is similar to other related protocols.

Considering the memory usage for a node in LEAP, let us assume that L stands for the number of units of memory required for storing F, D stands for the number of neighboring nodes, and each cryptography key requires 1 unit of memory storage. The memory usage at each node is:

Memory usage in LEAP= L+D+2D+1+1

1← Private key, 1← Group key, D← Cluster keys, 2D← Pair-wise keys, L←F function

In LEAP, each sensor node needs to keep 4 types of keys; private, pair-wise, cluster and group key. Each node thus requires the following number of units of memory for storage: 1unit for a single private key, 1 unit for a global group key, D units for D cluster head keys, 2D units for 2D pair-wise keys (2 pair-wise keys for each neighbor), and L units for its key chain (for the next key generation with F). Therefore, each node requires L+3D+2 units of memory [8].

Furthermore in the BROSK protocol each node requires the following number of units of memory for storage:

Memory usage in BROSK= D+D+1+1+1

1←Unique ID, 1←A nonce number (NA), 1←Group key, D← Session keys, D←Neighbors IDs

In BROSK, each sensor node needs to keep 1unit for a unique ID, 1 unit for a global group key, 1 unit for a nonce number which generate randomly, D unit for D session keys for communicate with its D neighbors by single hop or multi hop and D unit for D neighbors IDs storage. Therefore, each node requires 2D+3 units of memory.

In contrast, SKEW uses less memory at each node:

In a hierarchical WSN:

Memory usage = L+1+1+1+1.

1← Cluster number, 1← Group key, 1← Cluster key, 1← Key Version & ID, L← F function

In a distributed WSN:

Memory usage W= L+1+1+1.

1← Group key, 1←Unique ID, 1← Key Version, L← F function

In the hierarchical approach, each node requires 1 unit of memory for group key, 1 unit of memory for cluster key, 1 unit of memory for cluster number, 1 unit of memory for key version and sensor ID and L units of memory for key generator function. So, L+4 units of memory are required at each node in the hierarchical approach. Thus, memory usage at each sensor node in hierarchical approach in SKEW is far less than memory usage in LEAP.

In the distributed approach though, each node requires 1 unit of memory for group key, 1 unit of memory for unique ID, 1 unit of memory for key version and L units of memory for key generator function. So, L+3 units of memory are required at each node in the distributed approach. Thus, memory usage at each sensor node in distributed approach in SKEW is far less than memory usage in BROSK.

One of main challenges in wireless sensor networks is communication overhead that consumed 97% of sensor nodes energy [9]. So proposed protocol decrease the overhead by combine key refreshing and usual network messages.

If assumed network nodes transmit K messages for reply to a query that each message is N Byte and cryptography keys refreshed L times, so the nodes must send K*N+L*N Byte for it. So if S (S<N) Byte added to usual message body as refreshing message header in proposed protocol, transmitted rate decreases to (N+S)*L+ (K-S)*N= L*S+K*N. for example in a network sends a message periodically in each minute and message size is 32 Byte. If keys refreshed for 10 times in the network, so transmitted rate in one hour will be 32*60+32*10=2240 Bytes. Although this rate in proposed protocol with 8 Byte as message header decreases to 50*32+10*(8+32) =2000 Bytes.

Experimental results shown in Figure 11 describe the communication overhead improvement in proposed protocol than traditional protocols as well.

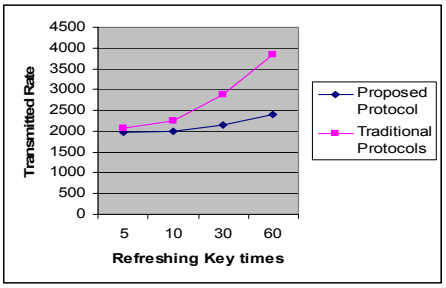


Figure 11: Transmitted Ratio improvement.

As shown above in proposed protocol when refreshing key time's increment, transmitted rate is far less than other traditional protocols and consummation energy for transmitting decreases too. Since information required for new key generation is piggybacked on transmitted messages, communication overhead for key distribution is reduced by 50% since key generation only entails 1 message communication in contrast to 2 messages in LEAP and BROSK.

In this paper implemented proposed protocol by visual Sense in a 200x200 dimensions environment with 100 sensor nodes which distributed randomly and each node has 10j initialize energy. Simulation time is 100 second that 5 second consumed for network configuration and energy model for nodes is 660 mj for transmit state, 390 mj for receive state and 22 mj for idle [27, 28].

We implement proposed protocol in both cases and compare the results with related protocols such as LEAP and BROSK in same status. Our implementation set up in two scenarios in first with two time's key refreshing and 5 times in finally. Experimental results shown in figure 12, 13 describe that the proposed protocol is better than related protocols in both cases for energy consummation. So its network lifetime will be better than other related protocols such as LEAP and BROSK.

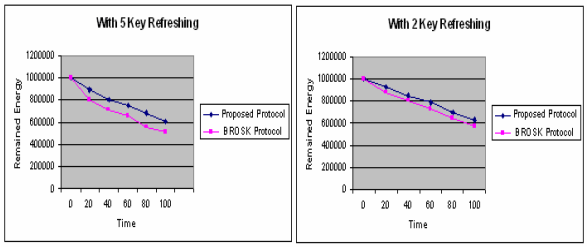


Figure 12: Remained Energy for Distributed Proposed Protocol.

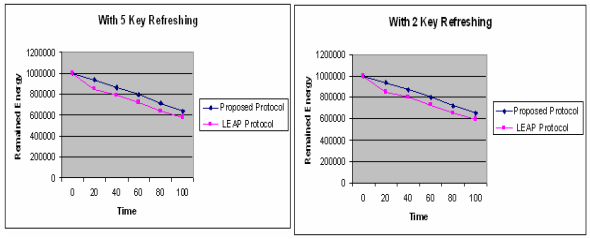


Figure 13: Remained Energy for hierarchical Proposed Protocol.

As shown above energy consummation in hierarchical protocols is more than distributed protocols by key refreshing increment. Therefore, our approach in both cases improvements energy consummation 20 to 25 percent.

Apart from communication issue, since keys are not directly transmitted within messages, i.e. only information about how they may be generated at the receiving end is transmitted; it is less probable that keys can be disclosed. Furthermore, since cryptographic keys such as cluster and group keys are refreshed periodically, disclosure of keys in a single period is less probable compared to when the keys remain fixed over the whole life of WSN. The number of generated keys in SKEW is comparatively low because only nodes that receive new encrypted messages must do refreshing. Therefore, SKEW entails less number of generated keys.

5. CONCLUSION AND FUTURE WORK

SKEW is a lightweight protocol for key management in WSNs. It tries to manage keys with minimum communication, key transmission and storage usage. It is a base key management protocol that preserves network security before start up. Other protocols can be mounted on top of this protocol. This protocol uses a refreshing mechanism to provide higher security. It does not require a specific key server for key broadcasting, and each node in each session can generate a key, and other nodes that

want to communicate with that node must update their keys. SKEW evaluation conclusion shown in Table 1, 2 in both cases, hierarchical and distributed.

Table 1: Distributed case conclusion.

	SKEW in Distributed Case	BROSK Protocol
Global Connectivity	All nodes	All nodes in key server radio range
Local Connectivity	By group key	By shared key
Resilience to sensor nodes capture	SKEW assumption	BROSK assumption
Scalability	√	√
Memory usage	**L+3**	**3+2D**
Communication overhead	**One** message for pair wise key, **one** message for group key	**two** message for pair wise key, **2D** message for group key
Energy consuming	2 key ≈ **3.7** j 5 key ≈ **4.1**j **for each node**	2 key ≈ **4.5** j 5 key ≈ **5.1j** **for each node**

Table 2: Hierarchical case conclusion.

	SKEW in Distributed Case	LEAP Protocol
Global Connectivity	All nodes	All nodes
Local Connectivity	By group key	By shared key
Resilience to sensor nodes capture	SKEW assumption	LEAP assumption
Scalability	√	√
Memory usage	**L+4**	**L+3D+2**
Communication overhead	**One** message for pair wise key, **one** message for group key	**two** message for pair wise key, **D** message for group key
Energy consuming	2 key ≈ **3.5** j 5 key ≈ **4j** **for each node**	2 key ≈ **4.4** j 5 key ≈ **5 j** **for each node**

Irrespective of all attributes realized and reported in this paper, a number of extensions to SKEW are in order:
We have introduced a lightweight protocol for WSN key distribution, using a simplified assumption about RAM. Removal of this assumption leads to heavier protocols. We are currently trying to find a similarly lightweight protocol without this assumption.

SKEW can be applied to unbounded networks with multiple or mobile base stations too. We are researching for the best approaches to propagate refreshing messages when there is no clustering.
Due to the self-key establishment attribute of SKEW, it can be applied to pervasive environments that require self-configuration.

If we assume time as resource, we can use SKEW for secure time synchronization in WSNs too.

In distributed SKEW architecture, all nodes which selected as cluster head communicate with the base station in a single hop. This can be extended to multi hop communication.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A Survey on Sensor Networks," IEEE Communications Magazine, Vol. 40, No. 8, 2002, pp. 102–114.

[2] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J. D. Tygar, "SPINS: Security Protocols for Sensor Networks," Seventh Annual International Conference on Mo-bile Computing and Networking (MobiCom 2001), 2001.

[3] D. Liu, and P. Ning, "Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks," 10th Annual Network and Distributed System Security Symposium, 2003, pp. 263-276.

[4] J. P .Walters, Z. Liang, W. Shi and V. Chaudhary, "Wireless Sensor Network Security: A Survey," LNCS Series: Signals and Communication Technology, 2006.

[5] Q. Huang, J. Cukier, H. Kobayashi, B. Liu and J. Zhang, "Fast Authenticated Key Establishment Protocols for Self-Organizing Sensor Networks," 2nd ACM international conference on Wireless sensor networks and applications, 2003, pp. 141-150.

[6] B. Lai, S. Kim and I. Verbauwhede, "Scalable Session Key Construction Protocol for Wireless Sensor Networks," IEEE Workshop on Large Scale Real-Time and Embedded Systems (LARTES), 2002.

[7] B. Dutertre, S. Cheung, and J. Leavy, "Lightweight Key Management in Wireless Sensor Networks by Leveraging Initial Trust," SRI-SDL-04-02, System Design Laboratory, 2004.

[8] S. Zhu, S. Setia and S. Jajodia, "LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," 10th ACM conference on Computer and Communications Security, 2003.

[9] M. Saraogi, "Security in Wireless Sensor Networks," ACM SenSys, 2004.

[10] O. Goldreich, S. Goldwasser, and S. Micali, "How to Construct Random Functions," Journal of the ACM, Vol. 33, No. 4, 1986.

[11] S. A. Camtepe, and B. Yener, "Key Distribution Mechanisms for Wireless Sensor networks: a Survey," TR-05-07 Rensselaer Polytechnic Institute Computer Science Department, 2005.

[12] D. Malan, M. Welesh, and M. Smith, "A Public Key Infrastructure for Key Distribution in Tiny Os Based on Elliptic Curve Cryptography," First IEEE International Conference on Sensor and Ad Hoc Communication and networks (SECONO4), 2004.

[13] G. Gaubatz, J. P. Kaps, and B. Sunar, "Public Key Cryptography in Sensor Networks," First European Workshop on Security in Ad Hoc and Sensor Networks, Heidelberg, Germany, 2004.

[14] J. Deng, R. Han, and S. Mishra, "Security, Privacy, and Fault Tolerance in Wireless Sensor Networks," Wireless Sensor Networks: A Systems Perspective, Artech House, 2005.

[15] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer Security Architecture for Wireless Sensor Networks," Second ACM Conference on Embedded Networked Sensor Systems (SensSys), 2004, pp. 162-175.

[16] D. Liu, P. Ning, and R. Li. Establishing, "Pair-wise Keys in Distributed Sensor Networks," ACM Trans., Inf. Syst. Secur., Vol. 8, No. 1, 2005, pp. 41–77.

[17] S. Rafaeli, and D. Hutchison, "A Survey of Key Management for Secure Group Communication," ACM Comput. Surv, Vol. 35, No. 3, 2003, pp. 309-329.

[18] H. Zhu, F. Bao, R. H. Deng, and K. Kim, "Computing of Trust in Wireless Networks," 60th IEEE Vehicular Technology Conference, Los Angeles, California, USA, 2004.

[19] J. Deng, R. Han, and S. Mishra, "INSENS: Intrusion-Tolerant Routing in Wireless Sensor Networks," CUCS-939-02, Department of Computer Science, University of Colorado, 2002.

[20] S. D. Mei, and B. Bing, "Review of Key Management Mechanisms in Wireless Sensor Networks," Acta Automatica Sinica, 2006.

[21] J. Hwang, and Y. Kim, "Revisiting Random Key Pre-distribution for Sensor Networks," ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04), New York, NY, USA, 2004.

[22] S. Camtepe, and B. Yener, "Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks," 9th European Symposium on Research Computer Security, 2004.

[23] M. Bohge, and W. Trappe, "An Authentication Framework for Hierarchical Ad hoc Sensor Networks," ACM workshop on Wireless Security, San Diego, CA, USA, 2003.

[24] J. Deng, R. Han, and S. Mishra, "Enhancing Base Station security in Wireless Sensor Networks", CU-CS-951-03, Department of Computer Science, University of Colorado, 2003.

[25] S. Slijepcevic, M. Potkonjal, V. Tsiatsis, S. Zimbeck, and M. Srivastava, "On Communication Security in Wireless Ad-hoc Sensor Network," Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'02), CMU, PA, 2002.

[26] J. Undercoffer, S. Avancha, A. Joshi, and J. Pinkston, "Security for Sensor Networks," CADIP Research Symposium, 2002.

[27] D. Carman, B. Matt, and G. Cirincione, "Energy Efficient and Low-latency Key Management for Sensor Networks," 23rd Army Science Conference, Orlando, FL, USA, 2002.

[28] M. Steiner, G. Tsudik, and M. Waidner, "Key Agreement in Dynamic Peer Groups," IEEE Transactions on Parallel and Distributed Systems, 2000.

[29] VisualSense. Available: http://ptolemy.berkeley.edu/visualsense.