

# Lightweight, Distributed Key Agreement Protocol for Wireless Sensor Networks

Che-Cheng Lin   Shiuhpyng Shieh   Jia-Chun Lin

*Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan*

*{jasonlin, ssp}@csie.nctu.edu.tw, kellylin@dsns.csie.nctu.edu.tw*

## Abstract<sup>1</sup>

*Secure key establishment is a fundamental security service in wireless sensor networks. It enables sensor nodes to secure their communications from malicious eavesdropping or tampering. However, due to restricted computing power and limited memory space, traditional public key cryptosystems may not be applicable to sensor hardware. To address this problem, we propose a lightweight distributed key agreement protocol, which enables each sensor node to establish shared secret keys efficiently with its neighboring nodes without the computation of modular exponentiation like in public key cryptosystems. Our protocol utilizes one-way hash function and bit-wise comparison operations, which are efficient and feasible for sensor hardware. The results showed that our approach provides better secure connectivity with lower redundant storage cost than existing schemes.*

## 1. Introduction

Traditional key exchange or key distribution protocols are based on infrastructural networks and rely on trusted third parties, such as the Key Distribution Centers. But the randomness of deployment and the ad-hoc fashion of communication make these protocols unsuitable for wireless sensor networks (WSNs). Moreover, the hardware constraints and the limited computing power of WSN make it infeasible to use the asymmetric (public key) cryptosystems on sensor nodes. This is because asymmetric cryptosystems requires high computation overhead which is not feasible for the low cost processors.

Symmetric key establishment is a possible solution for this problem, in which the encryption key is identical to the decryption key. Several schemes of symmetric key

establishment have been proposed recently to provide security service for wireless sensor networks. Most of these schemes pre-distribute (preload) a set of keys to each sensor node before deployment [3, 4, 5, 6, 7, 8, 9, 10]. Key pre-distribution is typically considered as an efficient way to establish symmetric keys; but it suffers from the collusion attack [11]. Moreover, these pre-distribution schemes require high deployment density to ensure reliable connection. This requirement reduces the variety of applications.

To address the above problems, we propose an innovative scheme for sensor nodes to establish symmetric keys with its neighbors. We designed a lightweight distributed key agreement protocol that mitigates the problems of existing key pre-distribution schemes.

The rest of this paper is organized as follows. Section 2 reviews the related work on symmetric key establishment and inter-sensor authentication for WSNs. Section 3 introduces the preliminary techniques used in our key agreement protocol. Section 4 gives performance evaluation and Section 5 concludes this paper.

## 2. Related Work

We categorize the reviewed symmetric key schemes into three categories: centralized keying scheme, random key pre-distribution scheme, and location-based schemes.

### 2.1 Centralized keying scheme

In centralized keying scheme, base station shares a key with each sensor node. Then the base station is involved in every establishment of inter-sensor key establishment. Schemes like SPINS [1] and SEKEM [2] belong to this category. However, Centralized keying schemes rely on the computing power of the base station, so this feature greatly increases overhead of the base station and also restrict scalability and flexibility of deployment.

---

<sup>1</sup> This work was supported in part by ITRI, Chung Shan Institute of Science and Technology, the International Collaboration for Advancing Security Technology (iCAST) and Taiwan Information Security Center (TWISC), under the National Science Council grants NSC96-3114-P-001-002-Y and NSC96-2219-E-009-013.

## 2.2 Random key pre-distribution

A set of keys is preloaded to sensor nodes before deployment in key pre-distribution schemes. After deployment, any pair of sensor nodes that share a common key are able to establish a secure connection. Eschenauer and Gligor [3] first proposed the key pre-distribution scheme for WSNs. In their scheme, each sensor node is preloaded a random subset of keys from a common key pool. Then any two nodes are able to find one common key to initiate a communication. Chan et al. [4] proposed two key pre-distribution approaches:  $q$ -composite key pre-distribution and random pairwise keys scheme. In  $q$ -composite key scheme, two sensor nodes have to share at least  $q$  common keys to establish a secure connection. The random pairwise keys scheme randomly chooses pairs of sensor nodes and assigns each pair a unique random key. Jeong et al. [5] proposed a key pre-distribution scheme which adopted random key pre-distribution scheme and double hash chain to reduce storage overhead.

Random key pre-distribution schemes provide a scalable approach for large-scale sensor networks with less communication overhead on key establishment. However, it limits the deployment of nodes, since these schemes require high enough deployment density to ensure the connectivity. Moreover, compromise of one node may expose the preloaded keys to adversaries and consequently weaken the security of sensor nodes that use these keys.

## 2.3 Location-based key pre-distribution schemes

In order to improve the connectivity and the security of random key pre-distribution schemes, many researchers proposed several location-based key pre-distribution schemes. In these schemes, reasonable amount of knowledge about the deployment is used to help pre-distribute the keys. PIKE [6] is proposed to improve the connectivity and the security of sparse networks. Liu et al. [7] proposed a practical deployment model, in which sensor nodes are deployed in group, and the nodes in the same group are close to each other after the deployment. Du et al. [8, 9] also developed a deployment model and proposed a key distribution scheme based on that. They made use of the deployment knowledge to avoid unnecessary key assignments on sensor nodes. Both performance and security are improved.

Location-based key pre-distribution schemes may improve the security of key pre-distribution schemes and reduce storage cost of sensor nodes. However, in large-scale sensor networks the sensor nodes may freely change their locations or sensor nodes are not deployed as expectation, these schemes not only incur even greater

overheads for indirect (multi-hop) key establishment but also reduce the connectivity with respect to the number of keys of the node.

## 3. Our protocol

The goal of this paper is to design a lightweight distributed key agreement protocol and make it practical and affordable for resource-limited sensor nodes. With our protocol, sensor nodes are able to generate symmetric keys with communicating parties and protect the sensitive or private information exchanged via public wireless medium. Our proposed scheme generates two “asymmetric” temporal keys for each node, that is, public middle key and private middle key. Communicating parties exchange their public middle keys and then derive the target secret key with each another’s public middle keys. The computation procedure of our key agreement protocol is totally different from conventional schemes, such as Diffie-Hellman key agreement protocol [12], which requires modular exponentiation computation and is too expensive for sensor hardware. Unlike conventional key agreement protocols, our proposed protocol does not need any modular exponentiation computation. Instead, we employed one-way hash function and bit-wise comparison operations to derive the secret keys.

We divide our key agreement protocol into three phases: initialization phase, key derivation phase, and key confirmation phase. Before detailing our proposed protocol, we give an overview by illustrating the scenario of key agreement session between Alice and Bob as shown in Figure 1.

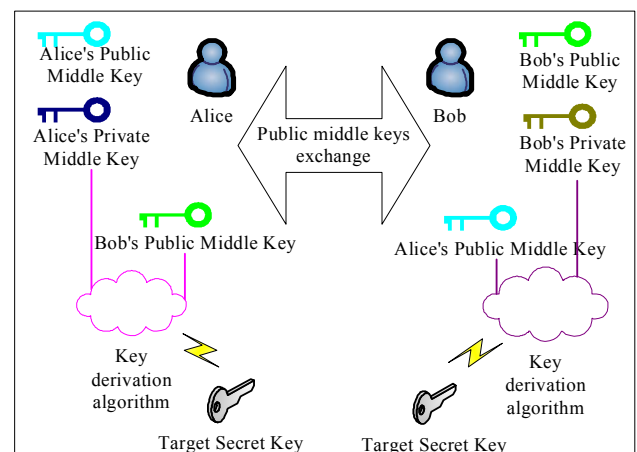


Figure 1. Overview of our protocol

In the initialization phase, Alice and Bob generate a pair of temporal keys: *private middle key* and *public middle key* respectively. In the key derivation phase, Alice and Bob exchange their public middle keys. Then Alice and Bob examine its own private middle key and the other's public middle key with a well-designed key derivation algorithm and derive a key called *target secret key*. Finally, in the key confirmation phase Alice and Bob perform a challenge-response mechanism to confirm that the target secret key they derive are the same or not. Table 1 lists the notations we use in our protocol.

**Table 1. Notations used in our protocol**

$A, B$	Principals
$b$	bit-length of target secret key
$n$	bit length of the secret number
${}^2s_A$	n-bit unsigned integer (secret number)
$SEC\_INT_A$	set of secret numbers
$PREFIX\_ONE(x)$	set of prefix bit-strings for every bit in $x$ whose value is 1
$PREFIX\_ZERO(x)$	set of prefix bit-strings for every bit in $x$ whose value is 0
${}^2P_A^1$	$PREFIX\_ONE({}^2s_A)$
${}^3P_B^0$	$PREFIX\_ZERO({}^3s_B)$
$p, q$	prefix bit-string elements in ${}^mP_i^j$
$H(\cdot)$	one-way hash function
${}^4C_A^1$	set of hashed prefix bit-strings
$PUB\_MK_A$	public middle key of principal $A$ .
$PRI\_MK_B$	private middle key of principal $B$
$SK_A$	target secret key derived by principal $A$
$K_i$	$i^{\text{th}}$ bit of the target secret key

### 3.1 Initialization phase

After two sensor nodes decide to launch a key agreement session, both nodes start the initialization phase. They employ one-way hash function and bit-wise operations to generate their public middle key and private middle key. All the data initialized in this phase will be eliminated from memory after the target secret key is derived.

The key-pair generation is divided into four steps as shown in Figure 2. In the first step, it randomly generates

$b$  pieces of  $n$ -bit unsigned integers which we called *secret numbers*:

$$SEC\_NUM_A : \langle {}^1s_A, {}^2s_A, {}^3s_A, \dots, {}^bs_A \rangle$$

Both public middle key and private middle key are computed from these secret numbers. Both bit-length  $b$  of target secret key and bit length  $n$  of the secret number are pre-defined based on the security strength needed.

In the second step, two sets of prefix bit-strings are computed from each secret number  $s_i$  as follows.  $PREFIX\_ONE(s_i)$  is the set of prefix bit-strings for every bit in  $s_i$  whose value is 1. On the other hand,  $PREFIX\_ZERO(s_i)$  is the set of prefix bit-strings for every bit in  $s_i$  whose value is 0, and  $\lambda$  is the prefix of the most significant bit ( $s_{i,n}$ ), where  $s_{i,j}$  is the value of the  $j^{\text{th}}$  bit of  $s_i$ .

$$PREFIX\_ONE(s_i) = \{s_{i,n}s_{i,n-1}\dots s_{i,j+1} \mid s_{i,j} = 1, j \geq 0\}$$

$$PREFIX\_ZERO(s_i) = \{s_{i,n}s_{i,n-1}\dots s_{i,j+1} \mid s_{i,j} = 0, j \geq 0\},$$

$$s_{i,n+1} = \lambda$$

Only bit-wise operations are required in the generation of these prefix bit-strings. To simplify our description, we use  $P_i^1$  and  $P_i^0$  to represent the two sets of prefix bit-strings:

$$PREFIX\_ONE(s_i) \rightarrow P_i^1 : \{p_1, p_2, \dots, p_v\}$$

$$PREFIX\_ZERO(s_i) \rightarrow P_i^0 : \{q_1, q_2, \dots, q_u\}$$

In the third step, we apply one-way hash function  $H$  to pair  $\langle {}^mP_A^1, {}^mP_A^0 \rangle$  and produce a pair of sets of hashed prefix bit-strings  $\langle {}^mC_A^1, {}^mC_A^0 \rangle$ .

$$\begin{aligned} {}^mP_A^1 = \{p_0, p_1, \dots, p_v\} &\xrightarrow{H(\cdot)} {}^mC_A^1 = \{H(p_1), H(p_2), \dots, H(p_v)\} \\ {}^mP_A^0 = \{q_0, q_1, \dots, q_u\} &\xrightarrow{H(\cdot)} {}^mC_A^0 = \{H(q_1), H(q_2), \dots, H(q_u)\} \end{aligned}$$

The composition of the hashed prefix bit-strings from  $P_i^1$  is an element of the public middle key, which is notated as  $C_i^1$ . Similarly, the composition of the hashed prefix bit-strings from  $P_i^0$  is an element of the private middle key, which is notated as  $C_i^0$ .

Finally, the public/private middle key pair is formed with the sets of hashed prefix bit-strings.

- *Public middle key*:  $PUB\_MK_A = \langle {}^1C_A^1, {}^2C_A^1, \dots, {}^bC_A^1 \rangle$

- *Private middle key*:  $PRI\_MK_A = \langle {}^1C_A^0, {}^2C_A^0, \dots, {}^bC_A^0 \rangle$

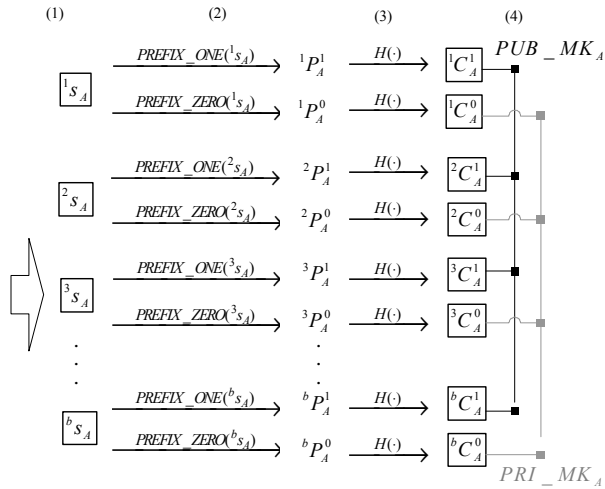


Figure 2. Initialization phase of key agreement

### 3.2 Key derivation phase

In key derivation phase, two communicating parties exchange their public middle keys and derive the target secret key by examining its own private middle key and the other party's public middle key. Let  $A$  and  $B$  are communicating parties and after the initialization phase  $A$  and  $B$  have their public/private middle key pair as follows:

$$A: \langle PUB\_MK_A, PRI\_MK_A \rangle$$

$$B: \langle PUB\_MK_B, PRI\_MK_B \rangle$$

Then  $A$  and  $B$  exchange their public middle keys via the wireless network. The delivered public middle keys could be authenticated with the inter-sensor authentication protocol [13].

$$A \rightarrow B: PUB\_MK_A = \langle {}^1C_A^1, {}^2C_A^1, \dots, {}^bC_A^1 \rangle$$

$$B \rightarrow A: PUB\_MK_B = \langle {}^1C_B^1, {}^2C_B^1, \dots, {}^bC_B^1 \rangle$$

Then two parties perform the key derivation algorithm to derive the target secret key. The main idea of our key derivation algorithm is that the relationship between two parties'  $m$ -th secret number determines the  $m$ -th bit value of the target secret key. Thus the key derivation algorithm compares the  $m$ -th secret number of two parties ( ${}^mS_A, {}^mS_B$ ) by examining the corresponding elements in the public middle keys ( ${}^mC_A^1, {}^mC_B^1$ ). Our algorithm employs a specific numeric comparison method, which compares the numeric scalar of two unsigned integers by

checking the prefixes of the original bit-string as the following equation.

$$(s_1 > s_2) = \begin{cases} \text{True,} & \text{if } PREFIX\_ONE(s_1) \cap PREFIX\_ZERO(s_2) \neq \emptyset \\ \text{False,} & \text{if } PREFIX\_ONE(s_1) \cap PREFIX\_ZERO(s_2) = \emptyset \end{cases}$$

We extended this comparison method by hashing the prefix bit-strings and checking the hash values to determine the relationship of original secret numbers: The sets of prefix bit-strings of the secret numbers are produced in the second step in the initialization phase.

$$PREFIX\_ONE({}^mS_i) \rightarrow {}^mP_i^1 : \{p_1, p_2, \dots, p_v\}$$

$$PREFIX\_ZERO({}^mS_i) \rightarrow {}^mP_i^0 : \{q_1, q_2, \dots, q_u\}$$

The sets of hashed prefix bit-strings are produced in the third step in the initialization phase:

$${}^mP_i^1 = \{p_0, p_1, \dots, p_v\} \xrightarrow{H(\cdot)} {}^mC_i^1 = \{H(p_1), H(p_2), \dots, H(p_v)\}$$

$${}^mP_i^0 = \{q_0, q_1, \dots, q_u\} \xrightarrow{H(\cdot)} {}^mC_i^0 = \{H(q_1), H(q_2), \dots, H(q_u)\}$$

If the one-way hash function is collision-free, then the relation between  ${}^mS_A$  and  ${}^mS_B$  could be derived by the following two equations.

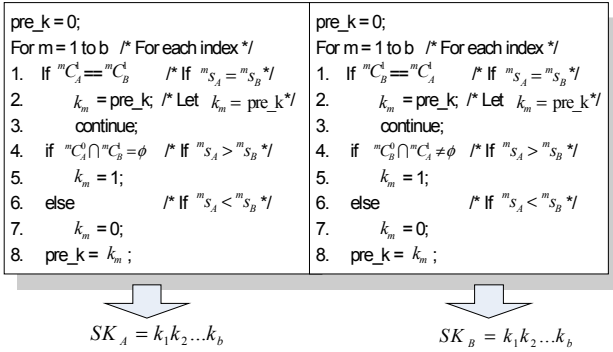
$$({}^mS_A > {}^mS_B) = \begin{cases} \text{True,} & \text{if } {}^mC_A^1 \cap {}^mC_B^0 \neq \emptyset \\ \text{False,} & \text{if } {}^mC_A^1 \cap {}^mC_B^0 = \emptyset \end{cases} \quad (1)$$

$$({}^mS_B > {}^mS_A) = \begin{cases} \text{True,} & \text{if } {}^mC_B^1 \cap {}^mC_A^0 \neq \emptyset \\ \text{False,} & \text{if } {}^mC_B^1 \cap {}^mC_A^0 = \emptyset \end{cases} \quad (2)$$

As shown in equation (1), the relationship between the two secret numbers is derived by examining the elements from  $A$ 's public middle key ( ${}^mC_A^1$ ) and the elements from  $B$ 's private middle key ( ${}^mC_B^0$ ). On the other hand, in equation (2), the relationship between the two secret numbers is derived by examining the elements from  $B$ 's public middle key ( ${}^mC_B^1$ ) and the elements from  $A$ 's private middle key ( ${}^mC_A^0$ ). Based on these two equations, we design two asymmetric key-deriving procedures for two parties in a key agreement session and those two procedures plays two roles in a key agreement session: one is the initiator and the other is the respondent. Figure 3 illustrates the pseudo code of the two procedures of key derivation. Assume that  $A$  is the initiator of the key agreement session. Both procedures contain a loop for examining the elements of the middle keys in order. The codes from line 1 to line 3 check the element from two parties' public middle key to see if two parties' secret numbers are equal. If the secret numbers are equal, let this bit value be the value of its previous bit. The codes from

line 4 to line 7 check each element of one's own private middle key and of the other's public middle key to get the result of secret number comparison. If the initiator's secret number is greater than respondent's at this round, the bit value is "1"; otherwise, the bit value is "0". After  $b$  rounds of checking, both parties derive their target secret key. Two parties have to confirm that they derive the same target key before using it. This is because the derived key could be different due to the collisions of the one-way hash function, so we need a confirmation mechanism to check the equality of the derived keys.

$$\begin{aligned} SEC\_NUM_A &: \langle {}^1s_A, {}^2s_A, \dots, {}^bs_A \rangle & SEC\_NUM_B &: \langle {}^1s_B, {}^2s_B, \dots, {}^bs_B \rangle \\ PRI\_MK_A &: \langle {}^1C_A, {}^2C_A, \dots, {}^bC_A \rangle & PRI\_MK_B &: \langle {}^1C_B, {}^2C_B, \dots, {}^bC_B \rangle \\ PUB\_MK_A &: \langle {}^1C_A, {}^2C_A, \dots, {}^bC_A \rangle & PUB\_MK_B &: \langle {}^1C_B, {}^2C_B, \dots, {}^bC_B \rangle \\ PUB\_MK_B &: \langle {}^1C_B, {}^2C_B, \dots, {}^bC_B \rangle & PUB\_MK_A &: \langle {}^1C_A, {}^2C_A, \dots, {}^bC_A \rangle \end{aligned}$$

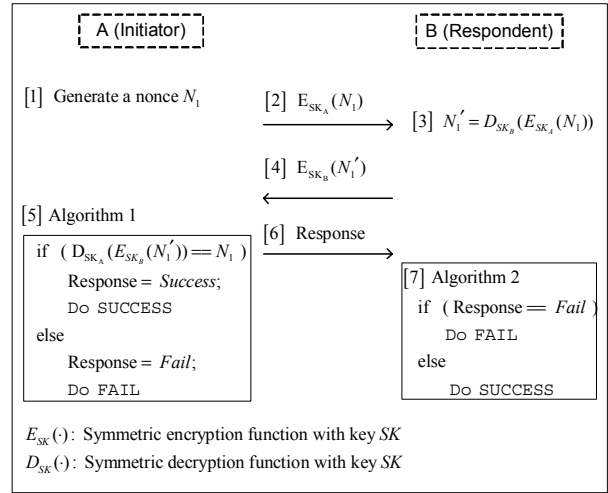


**Figure 3. Asymmetric procedures of key derivation**

### 3.3 Key confirmation phase

In this phase,  $A$  and  $B$  perform a challenge-response key confirmation. The key confirmation procedures for two parties are also asymmetric. Figure 4 illustrates the two confirmation procedures: one is for the initiator and the other is for the respondent. Here we assume that  $A$  is the initiator of the key-setup session.

The initiator  $A$  first generates a nonce  $N_I$ , and then encrypts it with its secret key  $SK_A$  which is derived in the previous phase and sends this encrypted nonce to  $B$ .  $B$  uses its secret key  $SK_B$  to decrypt the nonce and get  $N_I'$ . Then  $B$  encrypted  $N_I'$  with  $SK_B$  and sends it back to  $A$ .  $A$  derives  $N_I'$  by decrypting the message with  $SK_A$ , and then compares  $N_I'$  with  $N_I$ . If  $N_I'$  is not equal to  $N_I$ , it means that the key agreement session fails and  $A$  will send a failure message back to inform  $B$  the failure; if  $N_I'$  is equal to  $N_I$ , it means that the key agreement session succeeds and  $A$  will send a success message back to  $B$ , and complete the key agreement session. After a successful key agreement session, both  $A$  and  $B$  obtain the ability to protect mutual communication by various symmetric cryptosystems.



**Figure 4. Asymmetric procedures of key confirmation**

## 4. Evaluation

We evaluate our protocol according to the following generalized requirements and show the comparison result between our protocol and other key establishment or key pre-distribution schemes.

**Low computation cost.** Since our protocol utilizes one-way hash function and numeric comparing by prefix-checking, both only require constant time operations. In initialization phase, the generation of middle keys requires  $o(b \cdot n)$  hash operations; in key derivation phase, the derivation procedures require  $o(b \cdot n^2)$  bit-wise comparing operations. With the trade-off between security strength and computation cost,  $b$  and  $n$  are carefully chosen for the best cost effectiveness. As long as  $b$  and  $n$  are defined for the required security strength, the product of  $b$  and  $n$  or  $n^2$  will be considered as a large constant value. Therefore, our protocol which costs a constant amount of constant time operations is applicable in WSNs.

**Low storage requirement.** The key derivation phase may require a certain amount of memory space to store the middle keys. Though the order would be in the order of  $o(b \cdot n)$ , the memory usage are temporal and would be eliminated afterward. After the derivation, only the derived keys need being stored in the limited memory space. Therefore, the storage requirement is cost effective.

**Independent deployment.** Our protocol can work with any

kind of deployment configuration. As long as two sensor nodes are within each other's radio range, they can establish a secure key with our protocol.

**Mobility.** Sensor nodes may change their locations frequently. A relocated sensor node is able to establish new keys with its new neighbors as long as it could be authenticated; if not, inter-sensor authentication procedures need to be taken again before the key agreement protocol. The authentication of relocated sensor nodes will be considered in our future work.

**Scalability.** Our scheme supports large scale of deployment, since the overheads do not grow with the network size but the security strength.

**Robust to node compromise.** The compromised node exposes only the derived keys shared with its neighboring sensor nodes. It is possible to develop a revocation scheme based on the inter-sensor authentication protocol. With the revocation scheme, sensor nodes could drop the exposed keys that are shared with the compromised sensors. In this way, the security of the network hence remains intact.

Table 2 summarizes the comparison of our protocol and other key establishment or key pre-distribution schemes. The comparison is based on categories instead of each individual scheme. We examined these schemes in the aspects corresponding to the generalized requirements we mentioned above.

The major operations in DH or RSA key generation are modular exponentiation which requires plenty of multiplications, and therefore they are not feasible for sensor hardware. Centralized distribution (CD) [1, 2] scheme provides feasible and robust solutions for WSNs, and its storage requirement is cost-effective for that only the keys in use are saved in sensor nodes. However, the base station has to store all keys shared with each sensor node. Although base station is regarded as resourceful, the storage cost in base station will grow with the network size and restrain the scalability of this approach. Moreover, the mobility and connectivity are restricted to the area near the base station.

Random key pre-distribution schemes (RKP) [3, 4, 5, 8] and location-based pre-distribution schemes (LBP) [6, 7, 10, 11] provide scalable and lightweight approaches to construct secure connections among WSNs. Mostly, the secure connectivity relies on the deployment topology, this restriction reduces the variety of possible applications. Moreover, RKP and LBP pre-load a number of keys into sensor nodes may have the following problems: First, some keys would not be used throughout the network life, so it waste the storage space; second, if attacker compromises a sensor node and retrieves its storage, the preloaded keys are all compromised.

Our protocol mitigates these problems, and performs well in all the aspects of the evaluation. Table 3 also shows performance evaluation about the processing time. We use AMD Opteron 1.6Ghz processor as our platform and Linux 2.4.21 as the operating system. Obviously, our protocol costs the fewest processing time.

**Table 2. Comparison of our protocol and other schemes**

	DH, RSA	CD	RKP	LBP	Our Protocol
Major operations of key generation	Modular exponentiation	Symmetric encryption\ decryption	One-way Hash function	One-way Hash function	One-way Hash function
Storage cost	$\Theta(k)$	$\Theta(k)+1$	Predefined	Predefined	$\Theta(k)+1$
Deployment	Independent	Lightly Dependent	Medially Dependent	Heavily Dependent	Independent
Mobility	Yes	No	No	No	Yes
Scalability	Yes	No	Yes	Yes	Yes
Compromized link /compromised	$O(k)$	$O(k)$	$O(pk*nk)$	$O(pk*nk)$	$O(k)$

$k$  : number of keys in use.  $pk$  : number of keys preloaded.  $nk$  : number of nodes preloaded with the same key

**Table 3. Performance evaluation**

Algorithm	Processing Time
DH 1024	2.1 ms
LUCDIF 1024	3.94 ms
MQV 1024	2.19 ms
Our Protocol ( $b=1024$ )	0.06 us

## 5. Conclusion

In this article, we proposed a lightweight distributed key agreement protocol which utilizes one-way hash function and bit-wise comparison to securely establish the symmetric key between neighboring nodes in wireless sensor networks. Since both one-way hash function and bit-wise comparison are lightweight operations, our protocol is feasible for sensor hardware. Moreover, our protocol has the following advantages. First, it provides high connectivity of secure connection without preloading any redundant keys to sensor nodes. Second, it only needs low storage cost since no preloaded keys are required. Third, its computation cost is merely proportional to the security strength needed, rather than the network size. The proposed protocol is feasible for large scale sensor networks with mobile sensor nodes.

## 6. References

- [1] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen and D. E. Culler, "SPINS: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, 2002, pp. 521-534.
- [2] K. Jamshaid and L. Schwiebert, "SEKEN: secure and efficient key exchange for sensor networks," *IEEE International Conference on Performance, Computing, and Communications*, 2004.
- [3] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM conference on Computer and communications security*, 2002, pp. 41-47.
- [4] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of IEEE Security and Privacy Symposium*, 2003, pp. 197-213.
- [5] Y. S. Jeong, Y. C. Hwang, G. S. Kim, S. H. Lee, "Key Pre-distribution Scheme for Little Storage Space and Strong Security Strength in Large-Scale Wireless Sensor Network," *International Conference on Convergence Information Technology*, 2007, pp. 1572-1577.
- [6] H. Chan and A. Perrig, "PIKE: peer intermediaries for key establishment in sensor networks," in *Proceedings of IEEE International Conference on Computer and Communications Societies (INFOCOM)*, 2005, pp. 524-535.
- [7] D. Liu, P. Ning, and W. Du, "Group-based key pre-distribution in wireless sensor networks," in *Proceedings of the 4th ACM workshop on Wireless security*. Cologne, Germany: ACM Press, 2005, pp. 11-20.
- [8] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Trans. Inf. Syst. Secur.*, vol. 8, 2005, pp. 228-258.
- [9] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A Key Predistribution Scheme for Sensor Networks Using Deployment Knowledge," *Dependable and Secure Computing, IEEE Transactions on*, vol. 3, 2006, pp. 62-77.
- [10] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *Proceedings of IEEE INFOCOM*, 2004, pp. 586-597.
- [11] T. Moore, "A Collusion Attack on Pairwise Key Predistribution Schemes for Distributed Sensor Networks," in *Proceedings of IEEE PerCom Workshops*, 2006, pp. 251-255.
- [12] W. Diffie and M. Hellman, "Multiuser cryptographic techniques," in *AFIPS National Computer Conference*, Vol. 45, 1976, pp. 109-112.
- [13] W. H. Chen and Y. J. Chen, "A bootstrapping scheme for inter-sensor authentication within sensor networks," *Communications Letters, IEEE*, vol. 9, pp.945-947, 2005.