# A Resource-Optimal Key Pre-distribution Scheme with Enhanced Security for Wireless Sensor Networks

Tran Thanh Dai, Al-Sakib Khan Pathan, and Choong Seon Hong*

Networking Lab, Department of Computer Engineering, Kyung Hee University, Korea
{daitt, spathan}@networking.khu.ac.kr ,cshong@khu.ac.kr

**Abstract.** This paper proposes an efficient resource-optimal key pre-distribution scheme for providing improved security in wireless sensor networks. We exploit the advantages of two schemes with some modifications and combine our encoding scheme to get resource-efficiency and better security.

## 1 Introduction

In this paper, we present a new key pre-distribution scheme utilizing the advantages of two existing schemes [1] and [2]. Our analysis shows that our combined scheme performs better than each of these two schemes and ensures enhanced security than most of the other existing schemes. Principal contributions of our work are: (1) Considerable improvement in sensors' resource usage while keeping security as the top priority (2) Rigorous guarantee of successfully deriving pairwise keys that enable node-to-node authentication and (3) Better network resilience as compromising one or more pairwise keys does not influence the remaining pairwise keys

## 2 Our Proposed Scheme

The building blocks of our scheme are [1] (referred to as *LU decomposition scheme* here) and [2]. We modified both of the schemes to adapt them with our scheme.

**LU Decomposition Scheme [1].** Firstly, a large pool of keys P with size *s* is generated along with their identifiers. Then an $N \times N$ symmetric key matrix M is generated where *N* is the maximum number of sensor nodes that could be deployed. Each element $M_{ij}$ of M is assigned a distinct key from the key pool P such that $M_{ij} = M_{ji}, i, j = \overline{1, N}$. LU decomposition is applied to matrix M in the next step. The results of this decomposition are two matrices L, the lower triangular matrix and U, the upper triangular matrix; both of which are $N \times N$ matrices such that M = LU. Now, every sensor node is randomly assigned one row from the L matrix and one column from the U matrix, following the condition that, the *i*th row of L, $L_r(i)$ and the

* This work was supported by the MIC and ITRC projects. Dr. C. S. Hong is the corresponding author.

*i*th column of U, $U_c(i)$ always go together when assigned to a sensor node. Now, let us explain how a common key could be found between two nodes. Assume that sensor $S_i$ and sensor $S_j$ contains [$L_r(i)$, $U_c(i)$] and [$L_r(j)$, $U_c(j)$] respectively. When $S_i$ and $S_j$ need to find a common secret key between them for communication, they first exchange their columns, and then compute vector products as: $S_i$: $L_r(i) \times U_c(j) = M_{ij}$ and $S_j$: $L_r(j) \times U_c(i) = M_{ji}$. As M is the symmetric matrix, definitely, $M_{ij} = M_{ji}$. $M_{ij}$ (or $M_{ji}$) is then used as a common key between $S_i$ and $S_j$.

**Modified Blom's Symmetric Key Generation Scheme [3].** In this scheme, as long as no more than $\lambda$ nodes are compromised, the network is perfectly secure (this is referred to as the $\lambda$-secure property). Increasing $\lambda$ results in greater network resilience but also results in higher memory usage within each sensor node. During the pre-deployment phase, a $(\lambda+1)\times N$ matrix (where N is the maximum number of sensor nodes in the network) G over a finite field GF(q) and $\lambda$ (the security parameter discussed earlier) are constructed. G is considered as public information; any sensor can know the contents of G, and even adversaries are allowed to know G. In order to achieve the $\lambda$-secure property any $\lambda + 1$ columns of G must be linearly independent. Let $p$ be a primitive element of GF(q) and N < q. Then, each nonzero element in GF(q) can be represented by some power of $p$, namely $p^i$ for some $0 < i \le q-1$. A feasible G can be designed as follows:

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ p & p^2 & p^3 & \cdots & p^N \\ p^2 & (p^2)^2 & (p^3)^2 & \cdots & (p^N)^2 \\ & & & \vdots & \\ p^\lambda & (p^2)^\lambda & (p^3)^\lambda & \cdots & (p^N)^\lambda \end{bmatrix}_{(\lambda+1)\times N}$$

Before deployment, a random $(\lambda+1)\times(\lambda+1)$ symmetric matrix D over GF(q) is computed and used to compute an $N\times(\lambda+1)$ matrix A, which is equal to $A=(D.G)^T$, where $(D.G)^T$ is the transpose of $(D.G)$. Matrix D needs to be kept secret and should not be disclosed to adversaries or any sensor node. As D is symmetric, it is easy to see: $K = A.G = (A.G)^T$. Thus, $K_{ij} = K_{ji}$, where $K_{ij}$ is the element in K located in the *i*th row and *j*th column. In practice, G can be created by the primitive $p$ of GF(q). Therefore, when storing the *k*th column of G at node k, it is only necessary to store the seed $p^k$ at this node, any node can generate the column given the seed. After deployment, two sensor nodes $i$ and $j$ can find the pairwise key between them by exchanging their columns of G and using their private rows of matrix A to compute, $K_{ij} = K_{ji} = A(i).G(j) = A(j).G(i)$ where A(i) and G(j) represent the *i*th row of A and *j*th column of G respectively.

As we stated earlier that, our scheme is a combination of the two above-mentioned schemes with significant modifications so that it could be more apposite for the memory-constrained trait of wireless sensor networks. In our scheme, we only store the keying information in the sensor nodes, which are eventually used to derive two key halves to constitute a secret pairwise key. When two nodes want to communicate and are within the communication ranges of each other, the first key half is generated by the LU key decomposition scheme while the second half is generated by modified Blom's key computation scheme. The pairwise key used to secure the communication link between two nodes is derived based on these two halves, i.e., a concatenation of the two halves or using one way hash functions (Figure 1).

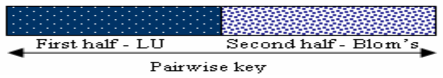

**Fig. 1.** Components of a pairwise key

Our scheme consists of mainly two phases:

**Prior Deployment: Key Distribution Phase.** Let us consider that each sensor node has a unique id $S_i$ where $i$=1,2,3…$N$. In our scheme, LU key decomposition scheme is applied first. After this step, each node $S_i$ contains one row from the L matrix and one column from the U matrix, respectively. While storing the row and column information, we apply an encoding scheme. As in each row and column there are two portions (first part non-zero and then possible zero element part), to store one row of L and one column of U in each sensor, we only store the nonzero portions of them and one value specifying the number of following zeros in the zero-element part. Thus, we store the keying information for the first half of the target key instead of storing the full length of the key. This storing method is very effective when the size of the network is very large. For generating the other-half of the key, other information are assigned using Blom's key generation scheme. So, the matrices G and A are generated offline. In this case, column G(j) is assigned to node j. Although G(j) consists of $(\lambda+1)$ elements, each sensor only needs to memorize only one seed, which could be used to regenerate all the elements in G(j). We also store the $j$th row of A at this node. This information is secret and should stay within the node. A(j) consists of $(\lambda+1)$ elements. Therefore, for this case, each node needs to store $(\lambda+2)$ elements in its memory. As the length of each element is similar to the length of the second key-half of the pairwise key, the memory usage of each node is $\frac{\lambda+2}{2}$ times the length of the key.

**After Deployment: Pairwise Key Establishment Phase.** After deploying the sensors on the area of interest (AOI), when two nodes (let $S_i$ and $S_j$) want to communicate with each other, they need to establish a pairwise secret key to transmit data securely. The steps that are followed for this are:

(1) $S_i$ and $S_j$ use some mechanisms to discover the location of the other (e.g., using query broadcast messages)
(2) $S_i$ and $S_j$ exchange messages containing $U_c(i)$, $U_c(j)$ from matrix U and a seed to generate, G(i), (G(j)) from matrix G
(3) $S_i$ and $S_j$ then compute the first half of the pairwise key as follows: $S_i : L_r(i) \times U_c(j) = M_{ij}$ and $S_j : L_r(j) \times U_c(i) = M_{ji}$
(4) $S_i$ and $S_j$ compute the second half of the pairwise key using the following equation: $K_{ij} = K_{ji} = A(i).G(j) = A(j).G(i)$

Up to this step, both $S_i$ and $S_j$ have the two halves of the pairwise key. To derive the pairwise key, the simplest way is to concatenate the first half with the second half. The pairwise key can also be created by using a one-way hash function. This key is stored in the memory of the two sensors for the rest of their communication. In our scheme, to reduce energy overhead for encryption and decryption of information exchanged between a pair of non-neighboring sensor nodes, each sensor node has a message relaying function. That is, messages exchanged between two non-neighboring nodes are encrypted and decrypted only by the concerned nodes using a common pairwise key established as described in the steps. The intermediate nodes only have to relay the messages to the receiving node. They do not need to understand the contents of the messages. Hence, they do not need to encrypt or decrypt the messages, which saves the computation and energy power.

## 3 Results and Conclusions

Here, we have presented a new pairwise key pre-distribution scheme for wireless sensor networks. Due to the page constraint, we have omitted the analysis part and shortened the details of our scheme. Our scheme is scalable because sensor nodes do not need to be deployed at a time rather they could be added later, and still they will be able to derive secret keys with existing nodes. Also the network-wide memory that is saved in our scheme could be used for network-wide distributed tasks. In Figure 2 we only show the storage efficiency that is gained from the encoding in our scheme.

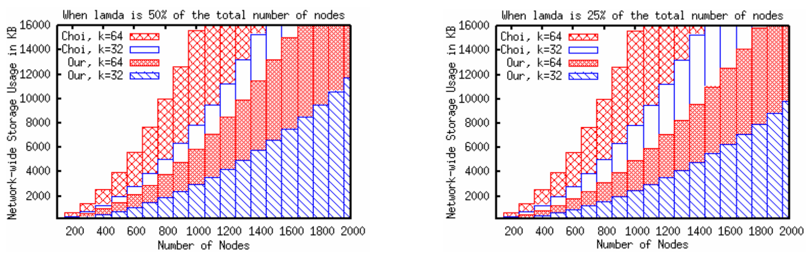

**Fig. 2.** Network-wide Memory usage in our scheme and [1] (left) if $\lambda$ is 50% of the total no. of nodes (right) if $\lambda$ is 25% of the total no. of nodes [$\lambda$ could be smaller]

## References

[1] Choi, S. and Youn, H., "An Efficient Key Pre-distribution Scheme for Secure Distributed Sensor Networks", EUC Workshops 2005, LNCS 3823 (2005) 1088-1097.

[2] Blom. R., "An optimal class of symmetric key generation systems", Advances in Cryptology: Proceedings of EUROCRYPT 84 (Thomas Beth, Norbert Cot, and Ingemar Ingemarsson, eds.), Lecture Notes in Computer Science, Springer-Verlag, 209 (1985) 335-338.

[3] Du, W., Deng, J., Han, Y. S., Varshney, P. K., Katz, J., and Khalili, A., "A Pairwise Key Predistribution Scheme for Wireless Sensor Networks", ACM Transactions on Information and System Security, Vol. 8, No. 2, May (2005) 228-258.