A Computationally-Efficient Construction for the Matrix-Based Key Distribution in Sensor Network*

Abedelaziz Mohaisen, Nam-Su Jho, and Dowon Hong**

Cryptography Research Team, Information Security Division
Electronics and Telecommunication Research Institute
Daejeon 305-700, Korea
{a.mohaisen,nsjho,dwhong}@etri.re.kr

Abstract. Key pre-distribution in wireless sensor network is aimed at delivering keys to sensor networks at the low expense of computation, communication, and memory while providing a high degree of security expressed by network resiliency to node capture. In this paper, we introduce a computationally efficient construction for the symmetric matrix-based key distribution. Particularly, this work introduces an original modification over the well known DDHV scheme (by Du et al.). Our modification shows that using a specific structures for the public matrix instead of fully random matrix with elements in $\mathbb{Z}_q$ can reduce the computation overhead for generating the key information and the key itself at the expense of small memory overhead. Our modification guarantees the same level of security for restricted network size. We show an extensive security analysis of the provided scheme in different settings and compare to the relevant works in the literature to demonstrate its merit.

Keywords: wireless sensor network, key distribution, computation efficiency, security.

1 Introduction

The security of wireless sensor network (WSN) is a challenging issue where both asymmetric (public) and symmetric key based algorithms are considered as possible solutions. However, because the public key based algorithms on the typical sensor nodes still require considerable amount of computation that is translated into processing time, symmetric key based algorithms that utilize the same key at the side of the sender and the receiver are favored for security the WSN. Particularly, these algorithms are shown to be computationally light and appropriate for sensors nodes. To use such algorithms in WSN, symmetric keys need to be distribution among the legitimate nodes in the network. However, because of the

* This work was supported by the IT R&D program of MIC/IITA. [2005-Y-001-05, Development of next generation security technology].

** Corresponding Author.

J.H. Park et al. (Eds.): ISA 2009, LNCS 5576, pp. 190–199, 2009.

WSN's frail infrastructure, traditional symmetric key distribution schemes that utilize key distribution centers (KDC) or trust third part (TTP) are obviously infeasible. To make the use of these algorithms in WSN possible, the concept of key pre-distribution (KPD) has emerged. In KPD, set of keys or keying materials are assigned to each node and used at the running time of the network to ensure secure communication. Several KPD schemes have been introduced in the literature for securing WSN. These schemes range from the graph-based cryptographic keys assignment such like the works by Eschenauer et al. in [1], Chan et al in [2], Hwang et al. in [3], Çamtepe et al. [4] and Mohaisen et al. in [5] to the more sophisticated online key generation schemes such like works by Du et al in [6], Liu et al. in [7] and [8], Mohaisen et al in [9] and [10], among others. In this paper, we review some of these schemes and provide a construction based on one of it to reduce its resources' consumption while maintaining the same level of security and connectivity.

Our original contribution in this article is a construction based on DDHV scheme [6] to reduce the used computation overhead at the expense of small communication and memory overhead. Our contributions therefore are summarized as follows: (1) We introduce a special construction for the public matrix used in [6] that reduces the computation overhead with a small additional communication and memory overhead which are yet comparable to other schemes. (2) We show a concrete evaluation for the soundness of the scheme, the security achieved and the resources evaluation. (3) To show a comparison between the modified DDHV scheme (OR-DDHV) and the original work, we introduce an extensive study that compares both schemes along with a few others from the literature with instantiated network scenarios and parameters.

The rest of this paper is organized as follows: section 2 introduces an overview of DDHV scheme followed by our scheme in section 3, section 4 introduces the analysis of both schemes where we show the overhead evaluation in terms of communication, computation and memory followed by the security analysis. Finally, section 5 draws a concluding remarks.

2 Overview of DDHV Scheme

The DDHV scheme in [6] utilizes Blom's linear construction in [11] with Eschenauer and Gligor's random key assignment concept in [1]. Both DDHV and Blom's schemes are based on the symmetry property of matrices to provide symmetric pairwise keys for the pairs of communicating nodes. DDHV scheme differs from the Blom scheme in that it utilizes multiple spaces for generating the key. In this paper, we will explain the discuss the symmetric matrix-based component of DDHVsince our modification is directly related to it. Also, modifications applied on the core of DDHV scheme can be utilized for the multiple space case.

Basically, a symmetric matrix of size $N \times N$ can be used for storing the different N^2 keys used for securing communication within the entire network of size N where each node s_i can have a row in that matrix. If two nodes s_i and s_j would like to communicate securely, they use the corresponding elements to

their identifier in their rows for encrypting and decrypting the communication traffic between them symmetrically. That is, in relation with the global matrix, the element E_{ij} is used by s_i and the element E_{ji} is used by s_j where both elements are equal because of the symmetry of the main matrix. To reduce the memory requirements, a linear algebraic-based construction is introduced where the size of matrices is determined by $\lambda \ll N$. Particularly, the following matrices are defined: a public matrix $\mathbf{G}$ of size $(\lambda+1)\times N$ and a private symmetric matrix $\mathbf{D}$ of size $(\lambda+1)\times(\lambda+1)$ where elements of $\mathbf{G}$ and $\mathbf{D}$ are randomly generated in the finite field $\mathbb{Z}_q$. Also, a matrix $\mathbf{A}$ is defined and computed as $\mathbf{A} = (\mathbf{DG})^T$ which is of size $N \times (\lambda+1)$. For any node s_i, the corresponding row $\mathbf{A}_r(i)$ from $\mathbf{A}$ and the corresponding column $\mathbf{G}_c(i)$ from $\mathbf{G}$ are selected and loaded in the node's memory. When s_i and s_j need to communicate securely, they exchange $\mathbf{G}_c(i)$ and $\mathbf{G}_c(j)$ respectively and then $k_{ij} = \mathbf{A}_r(i) \times \mathbf{G}_c(j)$ is computed in the side of s_i and $k_{ji} = \mathbf{A}_r(j) \times \mathbf{G}_c(i)$ is computed in the side of s_j. Obviously, the resulting keys are equal due to the symmetry property of the matrix $\mathbf{D}$.

To reduce the communication overhead, DDHV scheme introduced the a construction of $\mathbf{G}$ based on *Vandermonde matrix* which can be represented as in (1) where each node stores the corresponding field element in the matrix and generates the whole column from that value. To construct corresponding column from the given value, λ number of multiplications over $\mathbb{Z}_q$ are required. Similarly, to generate the key by multiplying $\mathbf{A}_r$ by $\mathbf{G}_c$, another λ number of multiplications over $\mathbb{Z}_q$ are required.[1]

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ s & (s^2) & (s^3) & \dots & (s^N) \\ s^2 & (s^2)^2 & (s^3)^2 & \dots & (s^N)^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s^\lambda & (s^2)^\lambda & (s^3)^\lambda & \dots & (s^N)^\lambda \end{bmatrix} \tag{1}$$

3 Modified Scheme DDHV Scheme (OR-DDHV)

Our modification for the DDHV scheme relies on reducing the computation overhead at the expense of additional communication and memory overhead while maintaining the same security level and connectivity. That is, we re-design the public matrix $\mathbf{G}$ so that to maximize the number of zeros in it while maintaining the linear independence between columns (or rows). Used columns or rows from our design will require less computation overhead to perform the inner product with a random row (or column) to generate a key.

Let the matrix in (2) be the typical $\mathbf{G}^T$ in which each row has only two nonzero values (a special type of orthogonal matrix in [12]). According to the DDHV scheme, each node has a column in $\mathbf{G}$ (i.e., row $\mathbf{G}^T$) in represented by

[1] Though a single seed can be used, an allotted memory space is still needed for storing a full column at the running time. To generate a key from the corresponding column and row costs same computation as of fully random column.

two non-zero values. Given the structure of $\mathbf{G}^T$, we define the offline and online phases in the following two sections.

$$\mathbf{G}^T = \begin{bmatrix} g_{11} & g_{12} & 0 & \dots & 0 & 0 \\ 0 & g_{22} & g_{23} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & g_{a_0 a_0} & g_{a_0 a_1} \\ 0 & g_{a_1 a_2} & 0 & \dots & 0 & g_{a_1 a_1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \end{bmatrix} \tag{2}$$

where $a_0 = \lambda - 1$, $a_1 = \lambda - 1$, and $a_2 = \lambda - 1$.

3.1 Offline Phase

The offline phase of the OR-DDHV scheme resembles the offline phase of the DDHV scheme to a great extent and consists of three steps shown in Fig. 1.

1. The administrator generates a symmetric matrix $\mathbf{D}$ of size $\lambda \times \lambda$ with elements in $\mathbb{Z}_q$ and the public matrix $\mathbf{G}$ of size $\lambda \times N$ with elements in $\mathbb{Z}_q$ where $\mathbf{G}$ satisfies the above restrictions.
2. The administrator computes $\mathbf{A} = \mathbf{G}^T\mathbf{D}$. The resulting $\mathbf{A}$ is of size $N \times \lambda$ and therefore its elements are in $\mathbb{Z}_q$.
3. For each node s_i, the administrator assigns the row with index i from the matrix $\mathbf{A}$ (e.g., $\mathbf{A}_r(i)$) and column with index i from matrix $\mathbf{G}$ (i.e. $\mathbf{G}_c(i)$).

Fig. 1. The offline phase of the OR-DDHV

3.2 Online Phase

The online phase of the OR-DDHV scheme is depicted in Fig. 2.

4 Analysis

In this section we analyze OR-DDHV scheme. We first introduce our insight on the network size's limitations. We then introduce a basic proof for the equivalence of the used keys followed by resources overhead and security analysis. Finally, we compare our scheme with other related works in literature.

4.1 Limitations on the Network Size

The maximum supported network size in our scheme is merely dependent on the parameters N and λ. In order to avoid a possible collision and maintain λ vectors (i.e., rows or columns) of $\mathbf{G}$ linearly independent, maximum network size is determined to $N = 2 \times \lambda$ for safety. Though, careful assignment for higher network size might satisfy collision-free criteria.

1. Firstly, two nodes s_i and s_j exchange their public columns $\mathbf{G}_c(i)$ and $\mathbf{G}_c(i)$ which can be represented as two non-zero values in $\mathbb{Z}_q$ and denoted as $g_{1i}, g_{2i}, g_{1i}, g_{2j}$.
2. In a vector $\mathbf{G}_c(j)$ with zero elements, the node s_i sets the received g_{1i} and g_{2j} from the node s_j with the identifier j into following positions in $\mathbf{G}_c(j)$:

$$\mathbf{G}_c(j)[j \mod \lambda] \leftarrow g_{1j}. \tag{3a}$$
$$\mathbf{G}_c(j)[(j+1) \mod \lambda] \leftarrow g_{2j}. \tag{3b}$$

3. Similarly, the node s_j reconstruct $\mathbf{G}_c(i)$ by plugging the received values g_{1i}, g_{2i} in the following positions:

$$\mathbf{G}_c(i)[i \mod \lambda] \leftarrow g_{1i} \tag{4a}$$
$$\mathbf{G}_c(i)[(i+1) \mod \lambda] \leftarrow g_{2i} \tag{4b}$$

4. The node s_i computes $k_{ij} = \mathbf{A}_r(i)\mathbf{G}_c(j)$.
5. The node s_j computes $k_{ji} = \mathbf{A}_r(j)\mathbf{G}_c(i)$.

Fig. 2. The online phase of the OR-DDHV

4.2 Equivalence of Keys

We can simply show that the generated key are equal. That is equivalent to showing that if $\mathbf{D}$ symmetric then $\mathbf{B} = \mathbf{G}^T\mathbf{D}\mathbf{G}$ is also symmetric and therefore the resulting keys are equal at both sides of s_i and s_j. To show the symmetry of $\mathbf{B}$, it is enough to demonstrate that $\mathbf{B} = \mathbf{B}^T$. That is, $\mathbf{B}^T = \left(\mathbf{G}^T\mathbf{A}\mathbf{G}\right)^T = \mathbf{G}^T\left(\mathbf{G}^T\mathbf{A}\right)^T = \mathbf{G}^T\mathbf{A}^T\mathbf{G} = \mathbf{G}^T\mathbf{A}^T\mathbf{G} = \mathbf{B}$. Since both k_{ij} and k_{ji} are elements in $\mathbf{B}$ which is symmetric, both keys are equal.

Let a_{ij}, d_{ij} and g_{ij} be the (i,j) elements in the matrices $\mathbf{A}, \mathbf{D}$ and $\mathbf{G}$ respectively. Also, let $\mathbf{A} = (\mathbf{D}\mathbf{G})^T$. From which we would like to show that $k_{ij} = A_r(i)G_c(j)$ and $k_{ji} = A_r(j)G_c(i)$ are equal.

We can write a_{ij} with corresponding to its multipliers as follows: $a_{ij} = (\sum_{k=1}^{\lambda} d_{ik}g_{ki})^T = (\sum_{k=1}^{\lambda} d_{ik}g_{ki})$ From which we can write

$$A_r(i) = [a_{1i}, a_{2i}, \dots] = \left[\sum_{k=1}^{\lambda} d_{1k}g_{ki}, \sum_{k=1}^{\lambda} d_{2k}g_{ki}, \dots\right] \tag{5}$$

Since $G_c(j) = [(g_{1j}, g_{2j}, \dots)]$, we can write $A_r(i) \times G_c(j)$ as follows:

$$A_r(i)G_c(j) = \sum_{l=1}^{\lambda}\left(\sum_{k=1}^{\lambda} d_{lk}g_{ki}\right)g_{lj} \tag{6}$$

Similarly, we can show that $A_r(j)G_c(i) = \sum_{l=1}^{\lambda}(\sum_{k=1}^{\lambda} d_{lk}g_{kj})g_{li}$. Now, we would like to check whether $A_r(j)G_c(i) = A_r(i)G_c(j)$ for any $i \neq j$. That is, we would like to show the following equality.

$$\sum_{l=1}^{\lambda}\left(\sum_{k=1}^{\lambda} d_{lk}g_{ki}\right)g_{lj} \stackrel{?}{=} \sum_{l=1}^{\lambda}\left(\sum_{k=1}^{\lambda} d_{lk}g_{kj}\right)g_{li} \tag{7}$$

By Taking the right side in (7) and change the index of the summations we get the that: $\sum_{l=1}^{\lambda}(\sum_{k=1}^{\lambda} d_{lk}g_{kj})g_{li} = \sum_{k=1}^{\lambda}(\sum_{l=1}^{\lambda} d_{kl}g_{li})g_{ki} = \sum_{k=1}^{\lambda}(\sum_{l=1}^{\lambda} d_{lk}g_{li})g_{ki}$.

Because $\mathbf{D}$ is symmetric, $g_{li} = g_{il}$, therefore the above can be rewritten as: $\sum_{l=1}^{\lambda} d_{l1}g_{lj}g_{1i} + \sum_{l=1}^{\lambda} d_{l2}g_{lj}g_{2i} + \cdots = (d_{11}g_{1j}g_{1i} + d_{21}g_{2j}g_{1i} + d_{31}g_{3j}g_{1i} + \dots) + (d_{12}g_{1j}g_{2i} + d_{22}g_{2j}g_{2i} + d_{32}g_{3j}g_{2i} + \dots) + (d_{13}g_{1j}g_{3i} + d_{23}g_{2j}g_{3i} + d_{33}g_{3j}g_{3i} + \dots) + \dots$. By resuming and arranging the terms we get the following:

$$g_{1j}\sum_{k=1}^{\lambda} d_{1k}g_{ki} + g_{2j}\sum_{k=1}^{\lambda} d_{2k}g_{ki} + \cdots = \sum_{l=1}^{\lambda} g_{lj}\sum_{k=1}^{\lambda} d_{lk}g_{ki} = \sum_{l=1}^{\lambda}\left(\sum_{k=1}^{\lambda} d_{lk}g_{ki}\right)g_{lj} \tag{8}$$

From (6) and (8), we get that (7) always holds. □

4.3 Resources Overhead

Communication overhead: The communication overhead required in the OR-DDHV scheme is $2\log_2 2^q = 2 \times q$ while it is q bits in the DDHV scheme when transferring a single field value from which the corresponding column in $\mathbf{A}$ is generated.

Computation overhead: The computation overhead in DDHV and OR-DDHV is two parts. The first part is required for reconstructing the public information from the field element and the second part is required for computing the inner product to generate the symmetric key.

- **Column's reconstruction computation:** The computation required in OR-DDHV scheme to reconstruct the corresponding column is negligible while it is λ number of multiplications in the field $\mathbb{Z}_q$ in DDHV scheme. That is, when λ is large, the number of computations over q will be also large. To illustrate how the reconstruction works for the case of DDHV scheme, given s^i, any element in the column is the result of multiplying the two previous elements. That is, $s^i = 1 \times s^i$, $(s^i)^2 = s^i \times s^i$ and so on.
- **Computation for inner product:** The computation for the inner product between the column from $\mathbf{G}$ and the row from $\mathbf{A}$ to obtain the symmetric key is 2 multiplications in our scheme since only two values are non-zero in $\mathbf{G}$'s corresponding column. On contrast, λ number of multiplications in the field $\mathbb{Z}_q$ are required in the case of DDHV scheme.

To sum up, the required computation overhead in term of multiplications in $\mathbb{Z}_q$ is 2 multiplications for OR-DDHV and 2λ multiplications for DDHV.

Table 1. Comparison between DDHV and OR-DDHV in term of the used resources where communication and memory are in bit per node and computation is in term of multiplications in the finite field $\mathbb{Z}_q$

Scheme	Communication	Computation	Memory
DDHV	$\log_2 q$	2λ	$(\lambda+1)\log_2 q$
OR-DDHV	$2\log_2 q$	2	$(\lambda+2)\log_2 q$

Memory overhead: Memory overhead is required for storing private and public information at each sensor. For storing the corresponding row in $\mathbf{A}$ for the node s_i, $\lambda \times q$ bits are required (for both of DDHV and OR-DDHV). However, our scheme requires $2q$ bit for storing its public information per node while DDHV scheme requires on q bits. Here we should emphasize on the fact the λ elements are to be stored for DDHV scheme at the running time of the algorithm after column reconstruction while our scheme requires space for 2 elements only.

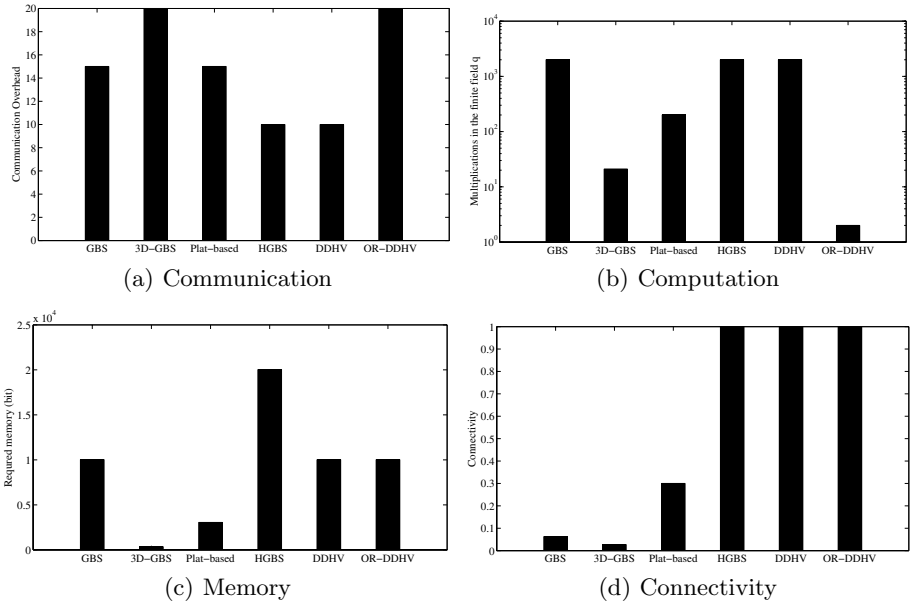
A summary of the comparison in terms of the required resources is shown in Table 1. Note that though the communication overhead in OR-DDHV is higher than in DDHV, it is still constant since q is fixed to accumulate the proper length of key. On the contrary, the computation in the OR-DDHV is constant while it increase linearly according to the security parameter λ in DDHV.

4.4 Security Analysis

The security analysis follows the analysis shown in DDHV and Blom works. That is, the system is λ-secure which means that an adversary needs to know λ number of different and linearly independent vectors (i.e., rows or columns) from the key generation construction to be able to know the keys between uncompromised nodes. Recall $\mathbf{G}$ in (2), $\mathbf{A}$, and $\mathbf{D}$ defined above. Also recall that a_{ij} and d_{ij} are the (i,j) elements of $\mathbf{A}$ and $\mathbf{D}$ respectively. Now we can define $\mathbf{A}_r(i)$ as $\mathbf{A}_r(i) = [\, a_{i1}\ a_{i2} \ldots\ a_{i\lambda}\,]$ where $a_{ij} = \left(\sum_{k=1}^{\lambda} d_{ik} g_{ki}\right)^T = \left(\sum_{k=1}^{\lambda} d_{ik} g_{ki}\right)$. The above $\mathbf{A}$ can be rewritten as:

$$\mathbf{A} = \begin{bmatrix} (g_{11}d_{11} + g_{12}d_{21})\ (g_{11}d_{12} + g_{12}d_{22}) \ldots \\ (g_{22}d_{21} + g_{23}d_{31})\ (g_{22}d_{22} + g_{23}d_{32}) \ldots \\ \vdots \qquad\qquad\qquad \vdots \qquad\qquad \ddots \end{bmatrix} \tag{9}$$

An adversary who would like to attack the above linear system must first reconstruct the proper $\mathbf{D}$. Since $\mathbf{D}$ is in $\mathbb{Z}^{\lambda\times\lambda}$, λ^2 number of linear equations are required for reconstructing it. That is, given that $\mathbf{G}$, the systematic structure of $\mathbf{A}$ and $\mathbf{G}$, and the symmetric property of $\mathbf{D}$ is publicly known information to the adversary, the adversary can obtain λ different linear equations by attacking a single node and reconstructing the different equations representing the row $\mathbf{A}_r(i)$. For instance, by attacking the nodes with an identifier 1, the attacker will have the $a_{11} = g_{11}d_{11} + g_{12}d_{21},\ a_{12} = g_{11}d_{12} + g_{12}d_{22},\ a_{13} = g_{11}d_{13} + g_{12}d_{23}, \ldots$.



(a) Communication (b) Computation

(c) Memory (d) Connectivity

Fig. 3. Comparison between our scheme and set of the related schemes: GBS, 3D-GBS, Plat-based, HGBS, and DDHV. For all of the above comparisons, $\alpha = 1$, $q = 10$, $N = 1000$.

By repeating the physical attack to λ different nodes, the adversary can construct λ^2 linear equation with λ^2 variables that can be solved to recover the whole private matrix $\mathbf{D}$ and construct any pairwise key between any pair of uncompromised nodes by just observing their public information. Note that the existence of multiple zeros in the $\mathbf{G}$ will not reduce the hardness of solving the above linear system since the different elements of the matrix $\mathbf{D}$ always exist in the resulting linear construction in $\mathbf{A}$. In DDHV scheme, however, all variables (represented by the different d's) appear in each equation rather than the two variables that appear in each equation in our construction.

4.5 Comparison with Related Works

In addition to the brief comparison between the OR-DDHV and the DDHV scheme shown in Table 1, we introduce a detailed comparison between the DDHV and a set of selected related works from the literature. Particularly, we compare the OR-DDHV to the grid-based scheme [8], 3D-GBS [8], Plat-based [10], HGBS [9], and DDHV [6]. The compared features are the communication (bit overhead), memory (bit storage), computation (multiplications over $\mathbb{Z}_q$), and connectivity. In DDHV and OR-DDHV, we set $\lambda = \alpha N$ to enable a fair comparison with other schemes. Also, for the connectivity alpha is made large enough so that the maximum possible connectivity is realized. Table 2 shows the detailed comparison.

Table 2. Comparison between our scheme and other schemes from the literature in terms of computation, communication, and memory

	Comm.	Memory	Computation	Connectivity
GBS	$\frac{3}{2}\log_2 N$	$\frac{3}{2}\log_2 N + (\alpha N + 1)\log_2 q$	$2\alpha N + 1$	$\frac{2}{N^{1/2}-1}$
3D-GBS	$2\log_2 N$	$2\log_2 N + 3(\alpha N^{1/3} + 1)\log_2 q$	$2\alpha N^{1/3} + 1$	$\frac{3}{N^{2/3}+N^{1/3}+1}$
Plat-based	$\frac{3}{2}\log_2 N$	$\frac{3}{2}\log_2 N + 3(\alpha N^{2/3} + 1)\log_2 q$	$2\alpha N^{2/3} + 1$	$\frac{3}{N^{1/3}}$
HGBS	$\log_2 N$	$\log_2 N + 2(\alpha N + 1)\log_2 q$	$2\alpha N + 1$	1
DDHV	$\log_2 q$	$(\alpha N + 1)\log_2 q$	$2\alpha N$	1
OR-DDHV	$2\log_2 q$	$(\alpha N + 2)\log_2 q$	2	1

To instantiate the above general comparison on a typical sensor network in order to measure the sensible merit of each scheme, we consider a network of size $N = 1000$, a field size $q = 10^2$, security parameter $\alpha = 1$, space for representing node identifier is $\log_2 1000 \approx 10$. Fig. 3 shows a comparison between the different schemes in the above features (i.e. communication, computation, memory, and connectivity). Particularly, Fig. 3(a) shows the comparison of communication overhead, Fig. 3(b) shows the comparison of computation overhead, Fig. 3(c) shows the comparison of memory overhead, and Fig. 3(d) shows the comparison of connectivity overhead. In these figures, we observe the following:

- While the advantage of the HGBS, DDHV, and OR-DDHV is the perfect connectivity at the expense of high memory consumption, the OR-DDHV scheme is the only one that provides such connectivity at the lower required computation.
- The computations provided in Fig. 3(b) are on the scale of $\log_{10}$. That is, other schemes require hundreds of time computation more than OR-DDHV scheme.
- Though the communication overhead of the OR-DDHV scheme is more than the required by the DDHV, this overhead is comparable to that of the HGBS at lower computations and memory overheads for same connectivity.

5 Conclusion

In this paper we introduced a construction for the matrix-based key predistribution scheme which is utilized in DDHV [6] and Blom's work [11]. We demonstrated that using an orthogonal matrix instead of fully random matrix with elements in $\mathbb{Z}_q$ as a public keying material will lead to a great reduction in the overhead represented by computation required for generating the key material and the key itself. While providing high connectivity, the introduced scheme has a comparable memory overhead to other schemes in the literature. In the

[2] Strictly stated, q must be large enough to accumulate reasonable key size. However, in this experiment $q = 10$ for all schemes to demonstrate relative advantage while not taking the key size into account.

near future, we will study the construction of other public matrices that maintain security of the system while being at resources feasible (memory, computation, and communication). As the introduced scheme limits the maximum number of nodes possible in the network to 2λ, we will investigate the cost of scalablity using the provided constructions.

Acknowledgment. The authors would like to thank Ku Young Chang and Ik Rae Jeong for their comments on earlier version of this work. We also would like to thank the anonymous reviewers for their comments.

References

1. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: ACM CCS, pp. 41–47 (2002)
2. Chan, H., Perrig, A., Song, D.X.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Security and Privacy, p. 197 (2003)
3. Hwang, J., Kim, Y.: Revisiting random key pre-distribution schemes for wireless sensor networks. In: SASN, pp. 43–52 (2004)
4. Çamtepe, S.A., Yener, B.: Combinatorial design of key distribution mechanisms for wireless sensor networks. IEEE/ACM Trans. Netw. 15(2), 346–358 (2007)
5. Mohaisen, A., Nyang, D., AbuHmed, T.: Two-level key pool design-based random key pre-distribution in wireless sensor networks. TIIS 2(5), 222–238 (2008)
6. Du, W., Deng, J., Han, Y.S., Varshney, P.K., Katz, J., Khalili, A.: A pairwise key predistribution scheme for wireless sensor networks. ACM Trans. Inf. Syst. Secur. 8(2), 228–258 (2005)
7. Liu, D., Ning, P., Sun, K.: Efficient self-healing group key distribution with revocation capability. In: ACM Conference on Computer and Communications Security, pp. 231–240 (2003)
8. Liu, D., Ning, P., Li, R.: Establishing pairwise keys in distributed sensor networks. ACM Trans. Inf. Syst. Secur. 8(1), 41–77 (2005)
9. Mohaisen, A., Nyang, D.: Hierarchical grid-based pairwise key predistribution scheme for wireless sensor networks. In: Römer, K., Karl, H., Mattern, F. (eds.) EWSN 2006. LNCS, vol. 3868, pp. 83–98. Springer, Heidelberg (2006)
10. Mohaisen, A., Maeng, Y., Nyang, D.: On the grid based key pre-distribution: Toward a better connectivity in wireless sensor networks. In: SSDU, pp. 527–537 (2007)
11. Blom, R.: An optimal class of symmetric key generation systems. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) EUROCRYPT 1984. LNCS, vol. 209, pp. 335–338. Springer, Heidelberg (1985)
12. Mohaisen, A., Hong, D.: Mitigating the ica attack against rotation based transformation for privacy preserving clustering". ETRI Journal 30(6), 868–870 (2008)