New Key Agreement Techniques for Sensor Networks

Abhishek Parakh and Subhash Kak

***Abstract*—We propose two computationally efficient key agreement algorithms. The schemes are ideally suited for computationally constrained environments such as sensor networks. The first proposed technique is general and uses matrix factorization. We provide constructive algorithms to implement the scheme. The second algorithm uses commutative property of matrices to distribute keys and provides two different keys per node pair. Both the algorithms are practical in terms of implementation, security provided and linear in computational complexity.**

***Index Terms*—Key distribution, sensor networks, matrix factorization**

I. INTRODUCTION

Sensor networks are becoming increasingly popular for applications such as patient health monitoring, detection of border crossings, bridge stress monitoring, signal relay points in battlefields and so on. In many of these applications sensors need to communicate securely to either relay data to base station or perform distributed computations. Therefore, encryption/decryption keys need to be distributed among the sensors.

Key distribution in sensor is particularly challenging because sensors have very limited computational power and transmission ranges. While in recent years the memory capacity for sensors has grown, they still cannot hold large number of keys for pair-wise communication. The key distribution challenge is further complicated by the fact that most sensors are deployed at random. As a result, we do not know a priori which sensors are going to be neighbors of other sensors that is within communication range of each other.

In general, for any key distribution scheme two techniques can be adopted - either install each node with pairwise symmetric keys before deployment or let sensors perform a public key exchange.

Installing pairwise symmetric keys is not a practical solution as it requires large storage capacity and does not allow for dynamic networking where nodes leave and new nodes join. This may happen because old sensors stop working and need to be replaced with new ones or the batteries run out.

If we consider a network to have N nodes, then a pair-wise symmetric key distribution would require each node to store $N-1$ unique keys (because of lack of a priori knowledge of sensor's neighbors). If AES is used as the encryption algorithm, this would require $(N-1)\cdot 128$ bits of storage as it is typical to have 10,000 sensors deployed in a network. If we allow for multiple sensor to use the same key, then we can reduce the number of keys installed on a given sensor, but that also means that once deployed there is a chance a sensor may not share a key with some of its neighbors. Therefore, if a sensor wished to communicate with a neighbor with which it does not share a key (or is out of its communication range), then link encryption (hop-by-hop) is used. In link encryption, assume sensor a wants to communicate with sensor d with which it does not share a key (or d is out of its communication range). If a shares a key with node b which in turn shares a key with node d, then a can send b a message such as $E_{k_{ab}}(m)$; where k_{ab} is a key shared between a and b. Node b upon receiving this message, first decrypts it and then re-encrypts it with key k_{bd} that it shares with node d and send it to d. This latter approach requires multiple encryption/decryptions along the way as well as a path finding and routing algorithm.

Eschenauer and Gligor [1] introduced the above approach where they assumed limited memory capacity and limited communication range for sensor networks. Further, they assumed random deployment of sensors, i.e. a sensor's neighbors were not known before deployment. As a result, after deployment the sensors performed a neighbor discovery in which they determined who their neighbors are and with which one of them they share keys. Then the sensors performed a path discovery to those sensors with which they do not share keys. Once a path was discovered, messages were sent using link-encryption. Although, the scheme proposed in [1] is very general and applicable to most scenarios, in practise one does have some knowledge of sensor neighborhood before deployment. Hence, EG requires the storage of larger number of keys on each sensor than may be required in a given scenario. Further, the path finding and routing protocols in a distributed sensor network are not trivial, especially when the number of neighbors one shares keys with are only a fraction of the number of neighbors actually in communication range.

Du et al. [2] assume deployment knowledge to reduce the number of keys stored per node. A gaussian probability distribution function is assumed with every sensor having a high probability of being deployed at a specific coordinate in a grid. However, such a scheme is not applicable to mobile nodes. Chan et al. [3] proposed a q-composite scheme that is similar to the EG scheme but requires that the nodes share q keys from the key ring instead of just one key and then final key to be used for encryption is computed as a function of these q shared keys.

In [4] it is assumed that mobile sensors handle the load of key distribution while static sensors only require minimal resources for key management. A bootstrapping technique is

Manuscript received December 10, 2014, revised March 10, 2015

A. Parakh is with the Nebraska University Center for Information Assurance, University of Nebraska, Omaha, NE 68182, e-mail: aparakh@unomaha.edu

S. Kak is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078

proposed in [5] that enables sensors to compute keys, once the network is deployed, based on network density, nodes memory and transmission range. A new post-deployment pairwise key distribution scheme for two-tier sensor networks is considered in [6] and a polynomial based key generating model is used for generating pair-wise keys to be shared with neighboring nodes.

The second approach, for key distribution, is that of using public key algorithms. In such algorithms, each sensor is installed with a public key and a corresponding private key. After deployment, the sensors broadcast their public keys to the neighboring sensors. The sending node can then encrypt all communication with the receiving node's public key. A number of public key algorithms have been implemented on sensors that claim to provide practical solutions, however, they consume many times more power than secret key encryption algorithms [7]. A hybrid scheme where public keys are used to exchange secret keys and the data encryption and transfer takes place using secret key algorithms is probably of a greater practical use as it reduces power consumption. One such hybrid approach is explored in [8] where the entire sensor network is divided into clusters managed by cluster heads. These cluster heads implement public key cryptography and aggregate data while individual sensors only use symmetric keys for encryption. A central key management server is used to establish keys in [9]. A hash chain based key distribution mechanism is discussed in [10].

Blom [11] discussed key exchange techniques based on the use of (n, k) linear codes with threshold property. A slightly modified version of Blom's algorithm was used by Du et al. [12] for establishing multiple shared keys between nodes by essentially executing Blom's scheme multiple times. Blundo et. al proposed a key distribution scheme [13] based on bivariate symmetric polynomial. Another scheme using LU Composition integrated with Elliptic Curve Diffie-Hellman has been proposed in [14]. Techniques based on polynomial interpolation and the idea of secret sharing are discussed in [15], [16] but have slightly higher computational cost compared to the proposed scheme. Similar polynomial based scheme for a two-tier network is proposed in [17].

In this paper we discuss an approach the bridges the gap between secret key and public key algorithms and enables sensors to establish shared secret keys with each other after deployment. In this approach each sensor is pre-installed with a small amount of seed information that can be exchanged with a neighbor to agree on a secret key. Any secret key encryption algorithm can be used to encrypt data thereon. Therefore, the proposed approach provides a number of advantages:

1) Pre-deployment of encryption keys is not required (key are generated after deployment).
2) Key agreement only has linear complexity.
3) A given node can share keys with all its neighbors resulting in larger connectivity within the network. This in turn leads to shorter path lengths compared to other methods where nodes share keys with only a fraction of its neighbors.
4) It allows for dynamic networks with nodes leaving and joining.
5) It assumes no pre-deployment knowledge of node locations and hence is general.

The proposed algorithm is based on matrix operations, where the computationally expensive pre-processing is pushed to pre-deployment phase and can be done at a base station.

In the following section, we present the proposed algorithm. In subsections II-A and II-B we discuss the size of matrices used and the complexity of the proposed algorithm. Section III discusses the security of the proposed algorithm for different size of matrices used and subsection III-C discusses the resilience against node capture. Section IV presents some constructive algorithms for the proposed scheme. Section V presents the second algorithm with commuting matrices and section VI concludes the paper.

II. PROPOSED ALGORITHM

Our aim is to provide alternatives to the use of public key algorithms for the establishment of shared secret keys. To achieve this we store a small amount of information, pre-deployment, on all the sensors. Once deployed, the sensors exchange a part of the pre-installed information with their neighbors to generate a shared secret key. This is very similar to what happens in Diffie-Hellman key exchange. However, here we do not require any exponentiation operation and the security of the scheme does not rely on the difficulty of mathematical operations (for example the security of Diffie-Hellman depends on the difficult of finding logarithms in finite fields). Since, the generation of the session key takes place after deployment, any sensor can perform a key exchange with any other sensor within its communication range. Larger connectivity between neighbors provides shorter path lengths through the network.

In the proposed algorithm all computations are performed modulo a large prime p. The proposed method is based on matrix factorization. It is further assumed that there are N sensors in the field and the deployment is done at random. The proposed algorithm consists of two phases - the pre-deployment phase and the key agreement phase.

The pre-deployment phase is performed by a base station. This phase essentially involves the factorization of a symmetric matrix. The symmetric matrix consists of random numbers from a finite field. These random numbers are the actual keys that will be used, therefore an appropriately large finite field must be used (usually on the order of 128 bits - if AES is being used). The base station performs the pre-deployment computations as follows.

Pre-deployment Phase (at base station):

1) Randomly choose a symmetric matrix K with elements in Z_p.
2) Find two matrices X and Y such that $XY = K$.
3) Randomly assign r^{th} row of X and r^{th} column of Y to each sensor node.

While distributing rows and columns, if node i receives the r^{th} row of matrix X, it also receives the r^{th} column of matrix Y. Here r is an integer chosen at random with uniform

probability from $[1, q]$, where matrix K is a symmetric matrix of size $q \times q$.

After every sensor is installed with a row-column pair from X and Y, the sensors can be deployed in the field using any mode of deployment. Once deployed, each sensor probes its neighborhood to discover the neighbors and then agree on a symmetric key as follows.

Key Agreement Phase:

When any two nodes, i and j, wish to agree on an encryption key, they exchange their columns of Y (in plaintext) and compute a common key as follows,

$$\text{Node } i \text{ computes: } K_{ij} = row_i(X) \cdot col_j(Y)$$

$$\text{and node } j \text{ computes: } K_{ji} = row_j(X) \cdot col_i(Y)$$

As matrix K is symmetric, $K_{ij} = K_{ji}$. Since a node in the network has only one row and one column installed on it, the notation $row_i(X)$ denotes the row of X that was stored on node i. This must not be confused with row i of X. Similarly, $col_j(Y)$ is the column of Y assigned to node j.

Fig. 1 show the pictorial representation of matrices X and Y and fig. 2 illustrates a sensor network in which all the nodes within the communication range with each other can share an encryption key. Only a few nodes with their communication ranges are shown.

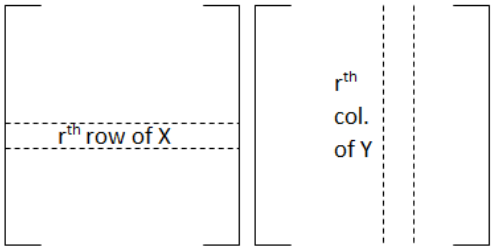

Fig. 1. A node gets the r^{th} row and column of matrices X and Y.

A. Size of Matrices

Assume a network with N nodes. If we wanted every node-pair in the network to share a randomly and uniformly chosen key, then there would exist $\frac{N(N-1)}{2}$ independent keys. In other words, if an eavesdropper is able to determine the key being used for the link between nodes i and j, then he gains no advantage in determining the key being used on any other link. For this to be true, the symmetric key matrix K needs to be of size $N \times N$ since the upper (or lower) triangle of the matrix contains $\frac{N(N-1)}{2}$ elements (not including the diagonal elements that only form "self-keys").

A $N \times N$ key matrix can be factored into X and Y in different ways where the size of matrix X is $N \times m$ and the size of matrix Y is $m \times N$. As a result, every node in the network can receive a unique row-column pair during the pre-deployment phase. A simple row-column pair distribution algorithm would give node i, the i^{th} row of X and i^{th} row of Y. As a result, the storage required on each node is $2m$ integers.

In general if reuse of keys is allowed, matrix K may be of size $q \times q$, where $q \leq N$, and matrices X and Y are of sizes $q \times m$ and $m \times q$, respectively. In this case, step 3 of the algorithm randomly assigns rows and columns, where a row-column pair may go to more than one sensor node. This implies that some of the node pairs may share the same encryption key. More precisely, a $q \times q$ matrix has $\frac{q(q-1)}{2}$ random and independent numbers. Therefore, it is expected that any given key will be shared by $\frac{N(N-1)}{q(q-1)}$ links and each row-column pair may go to $\frac{N}{q}$ nodes.

B. Linear Computational Complexity of Key Generation

The key generation operation for a given link involves the multiplication of one row of X with a column of Y. If we assume the size of X is $q \times m$ and the size of Y is $m \times q$, then computing a key requires m multiplications and $m - 1$ additions. Further this is dependent on the size of matrices X and Y which in turn depends on the desired level of security. In the worst case X and Y are of size $N \times N$ and hence key generation requires N multiplications and $N - 1$ additions.

III. Security of the Proposed Scheme

A. Size of K is $N \times N$

Assume that matrix K is of size $N \times N$ and therefore X and Y are of sizes $N \times m$ and $m \times N$ respectively. The upper triangle (including the diagonal) of matrix K has $\frac{N(N+1)}{2}$ elements all of which are generated from $2(N \times m)$ elements. For each key there are p possibilities and it is clear that in the absence of any knowledge of the elements of X and Y, all the p possibilities for every key remain equally likely.

However, since the columns of Y are being transmitted in plain text, an eavesdropper can record these columns. Further, if the eavesdropper is able to listen to N distinct transmissions, of columns of Y, from N distinct nodes, then there remain $N!$ possibilities to arrange these columns in matrix Y. However, this gives no information about matrix X which has $N \times m$ elements in it.

Now, if the adversary tries to guess the values of elements in matrix X, then every new row of X gives the adversary decreasing amount of information. This is because the key matrix K is symmetric. As a result, the first row of X when multiplied with Y gives $N - 1$ unique keys, the second row of X when multiplied with Y gives $N - 2$ unique keys and so on. However, to determine all the keys this way, the adversary will have to determine all the rows of X, i.e. $N \times m$ values from Z_p.

Two possible choices remain for the eavesdropper:

- If $m > N$ then it is more difficult to determine X than it is to determine the elements of K directly. Therefore, capturing N columns of Y gives no advantage.
- However if $m < N$, then determining the first row of X (m values) will result in $N - 1$ keys, determining the second row X (another m values) will result in $N - 2$ keys and so on. Consequently, an adversary can stop

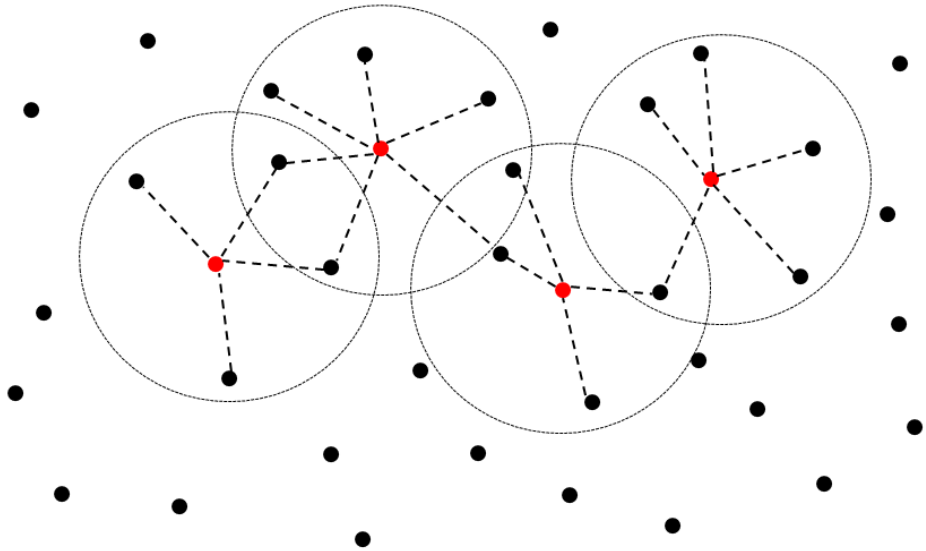

Fig. 2. Illustration of a network. We've shown the communication ranges with dotted circles for a few nodes (shown in red at the center of the circles). The dashed lines between nodes represents that a key agreement will take place between these nodes since they are within communication range of the red nodes. Similarly every node will have its own communication range and it will agree on a key with all nodes in its communication range. Nodes can join and leave the network at will.

determining the rows of matrix X when determining m values of that row gives fewer keys than m. At this point it would be beneficial for the adversary to directly guess the remaining values in the key matrix K.

This is in addition, however, to determining which of the $N!$ ways are the columns of matrix Y arranged and for each arrangement different possible keys exist.

Moreover, such an attack may be impractical because it requires an adversary to listen to every transmission of Y that takes place in the *entire* network.

B. Size of K is $q \times q$, $q < N$

Assume that matrix K is of size $q \times q$, $q < N$, then the matrices X and Y are of size $q \times m$ and $m \times q$. Since, there are fewer than N rows and columns, the rows-columns pairs pre-loaded on the sensors can be pre-loaded randomly with repetition. As a result, some of the links in the network will share the same encryption key.

The probability that any two nodes will share the same key may be computed as follows. Note there are two possible events that can occur.

1) Two node pairs are assigned the same K_{ij} from the key matrix.
   This may happen because $q < N$ and same keys are used more than once. Since there are a total of $\frac{q(q-1)}{2}$ possible keys, the probability that two node pairs will be assigned the same key is: $\frac{1}{\frac{q(q-1)}{2}} = \frac{2}{q(q-1)}$.
2) Two node pairs are assigned different K_{ij}s from the key matrix.
   This happens with a probability of $1 - \frac{2}{q(q-1)}$. However, since each K_{ij} is randomly and uniformly picked from Z_p, any two elements of matrix K will be equal with probability $\frac{1}{p}$.

Therefore, the total probability that any two node pairs will receive the same encryption key is given by:

$$(1 - \frac{2}{q(q-1)}) \cdot \frac{1}{p} + \frac{2}{q(q-1)} \cdot 1$$

where $q > 1$. When $q = 1$ the probability of repetition of the same key is 1 and refers to a master key system.

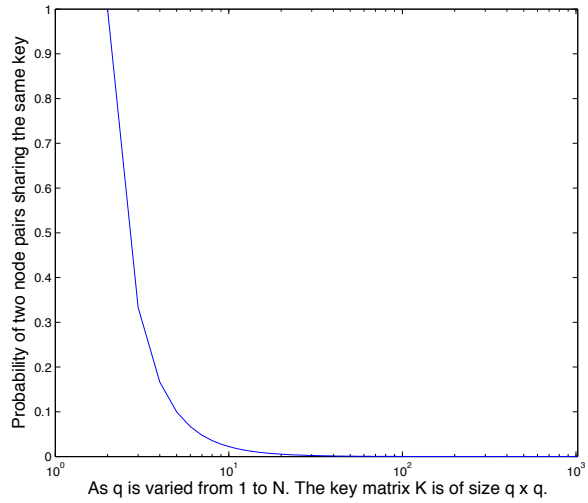

Fig. 3. Probability of two node pairs sharing the same key as q is varied from 2 to N. Size of the key matrix K is $q \times q$.

Figure 3 shows how the probability of sharing the same key between two node pairs decreases as size of the key matrix K is increased. We see that at size of $q = 200$ the probability of

sharing a key decreases to less than 0.0001. The network size is fixed at $N = 1024$.

C. Resilience Against Node Compromise

Compromise of nodes leads to greater loss of information than eavesdropping. However, unlike Blom's scheme [11], the proposed algorithm does not possess a threshold property. Hence, the degradation of network characteristics is graceful with compromise of nodes. Further, depending on how the factors of K are computed the scheme can be adapted to possess a threshold property if desired.

If the size of K is $N \times N$ and each node receives a unique row-column pair then if an attacker was to compromise m nodes, he will be able to construct: $(N-1)+(N-2)+\ldots+(N-m) = m \cdot N - \frac{m(m+1)}{2}$ keys (elements) from matrix K.

If the size of K is $q \times q$, $q < N$ then the probability of construction of all the keys upon L node compromises will be based on the retrieved L row-column pairs. However, since $q < N$ not all of these row-column pairs are distinct. The minimum number of nodes that an adversary will need to compromise in order to retrieve q distinct row-column pairs is q nodes. Consequently, for L nodes compromised the probability that the adversary will see q distinct row-column pairs at least once is computed as follows:

Suppose in L compromises the adversary does not see all the q distinct row-column pairs, i.e. at least one pair is missing. In other words, we can compute all the ways in which L choices (with repetition) can be made from $q-1$ possible pairs. Out of total q pairs there is $\binom{q}{(q-1)}$ ways to choose $q-1$ pairs. From each of these possibilities, we can make L random choices with repetition in $(q-1)^L \cdot \binom{q}{(q-1)}$ ways. However, we need to subtract the double counted possibilities which is $q^2 - 2q$. As a result, the total number of ways, an adversary will fail to see all possible row-column pairs in L compromises is given by $(q-1)^L \cdot \binom{q}{(q-1)} - (q^2 - 2q)$. This is out of the total number of possible ways all choices can be made, i.e. q^L. The probability that an adversary will successfully see all the pairs in L compromises is then given by,

$$1 - \frac{(q-1)^L \cdot \binom{q}{(q-1)} - (q^2 - 2q)}{q^L}$$

IV. Constructive Algorithms to Determine X and Y

Although in general X and Y may be chosen by trial and error, their determination becomes easier if one of the following methods is used. The examples below present some of the different methods to construct X and Y.

1. One method would be to choose Y random and non-singular and compute $X = K \cdot Y^{-1}$.

For example, assume a symmetric matrix K of size 3×3 and we work mod 11.

Let $K = \begin{bmatrix} 3 & 4 & 6 \\ 4 & 5 & 2 \\ 6 & 2 & 1 \end{bmatrix}$ and $Y = \begin{bmatrix} 3 & 4 & 8 \\ 1 & 5 & 2 \\ 9 & 10 & 4 \end{bmatrix}$

Then $Y^{-1} = \begin{bmatrix} 0 & 6 & 8 \\ 2 & 4 & 5 \\ 6 & 4 & 0 \end{bmatrix}$ and

$X = \begin{bmatrix} 0 & 3 & 0 \\ 0 & 8 & 2 \\ 10 & 4 & 3 \end{bmatrix}$

2. Another example of construction of X and Y where they are smaller than the size of K is as follows (again working mod 11).

Let $X = \begin{bmatrix} 1 & 3 & 1 \\ 3 & 4 & 1 \\ 9 & 9 & 1 \\ 4 & 6 & 1 \\ 5 & 1 & 1 \end{bmatrix}$ and

$Y = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 5 & 3 & 2 & 4 \\ 1 & 3 & 9 & 4 & 5 \end{bmatrix}$ then

$K = \begin{bmatrix} 5 & 8 & 8 & 0 & 7 \\ 8 & 4 & 2 & 4 & 2 \\ 8 & 2 & 1 & 9 & 6 \\ 0 & 4 & 9 & 9 & 0 \\ 7 & 2 & 6 & 0 & 3 \end{bmatrix}$

Here every node stores $2m = 2 \cdot 3 = 6$ elements from Z_p.

3. LU factorization may be used.

4. Computing powers of matrices. Assume that matrix K is diagonalizable; then $K = M^{-1}AM$ where M is a matrix whose columns are the eigenvectors of K and A is a diagonal matrix of eigenvalues of K. With such a factorization the algebra on K reduces to the algebra on the elements of the diagonal matrix A. For example $K^r = M^{-1}A^rM$ and since A is a diagonal matrix $A^r = (a_1^r, a_2^r, \ldots, a_q^r)$ where a_i are the diagonal elements of A. Then we may factor K as follows:

1) Randomly choose a symmetric diagonalizable matrix K with elements from the finite field Z_p.
2) Randomly choose a element r from the field and compute $X = K^{\frac{1}{r}}$ and $Y = K^{1-\frac{1}{r}}$.

If K is diagonalizable then the r^{th} root can be computed as discussed above.

In this method, if an adversary captures all the columns of Y and figures out the layout of the captured columns of Y in the matrix, then to compute X, he will have to compute the $(r-1)^{th}$ root of the matrix Y. However, not knowing the value of r which was randomly and uniformly chosen from Z_p there are $p \cdot (r-1)$ possible choices for X.

V. Using Commuting Matrices

If one was to use commuting matrices the requirement of matrix K being symmetric can be eliminated. This would require every node to store some additional information that provides every node pair two different keys that are used for communication depending on which node initiates the communication. These keys may be hashed together to generate another key that is used for encryption thus further improving the security of the system. The algorithm works as follows:

Pre-deployment

1) Choose two $q \times q$ matrices X and Y such that $XY = YX$ and Y is symmetric.
2) Randomly pick r from a uniform distribution over $[1, q]$.
3) Assign node i the r^{th} row and column of X and the r^{th} column of Y.

Assume two nodes i and j wish to agree on a key then the key agreement proceeds as follows,

1) Node i sends its i^{th} column of Y to node j.
2) Node j sends its j^{th} column of Y to node i.
3) Node i computes $K_{ij} = row_i(X) * col_j(Y)$ and node j computes $K_{ij} = col'_i(Y) * col_j(X)$.
4) Node i computes $K_{ji} = col'_j(Y) * col_i(X)$ and node j computes $K_{ji} = row_j(X) * col_i(Y)$.

Where $col'_i(Y)$ is the column of Y transposed. Figure 4 illustrates the predistribution of rows and columns for node i.

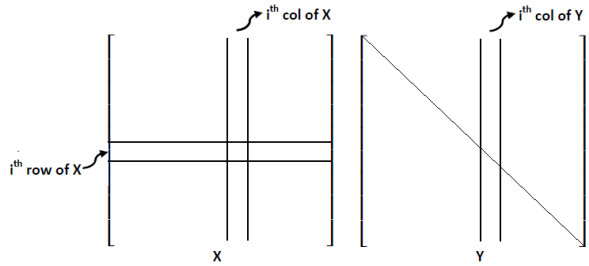

Fig. 4. Predistribution of rows and columns for node i for the case when $XY = YX$ and Y is symmetric. The line across the diagonal of matrix Y indicates the symmetry.

Finding Commuting Matrices X and Y:

Matrix diagonalization may be used to find two such matrices using the following steps:

1) Choose a diagonalizable symmetric matrix Y at random.
2) Diagonalize Y such that $Y = M^{-1}D_yM$, where D_y is a diagonal matrix with eigenvalues of Y.
3) Randomly pick a diagonal matrix D_x and compute $X = M^{-1}D_xM$.

The above algorithm generates two matrices that commute with each other as seen below,

$$XY = M^{-1}D_xMM^{-1}D_yM = M^{-1}D_xD_yM$$

and

$$YX = M^{-1}D_yMM^{-1}D_xM = M^{-1}D_yD_xM$$

Since D_x and D_y are diagonal matrices $D_xD_y = D_yD_x$. If an eavesdropper is able to listen to q distinct columns of Y being transmitted and also determine which out of $q!$ ways they are to be arranged then he can diagonalize Y and retrieve M. In order to reconstruct all the keys in the network he then has to guess the values in the diagonal matrix D_x and there are p possible values for every eigenvalue in D_x.

Unlike the previous algorithm, using commuting matrices requires X and Y to be square matrices. However, the size of the matrices may be $q \times q$ where $q \leq N$. The advantage of using commuting matrices is that every node pair now shares two keys that may be used for encryption in following ways:

1) For fast encryption and low computational overhead, encryption keys are used to seed random number generators. Then to encrypt data the sequence of random bytes generated (from the RNG) is XORed with the bytes of the data and transmitted. However, if the same seed is used in the RNG for data sent both ways, i.e. from node i to j and vice versa, then it can lead to a two-time-pad attack. As a result, the two encryption keys generated: K_{ij} can be used to seed the RNG for encrypting data sent from node i to j and K_{ji} can be used for data sent from j to i.
2) If two-time-pad is of no concern then we could $K = Hash(K_{ij}||K_{ji})$ as the common encryption key between nodes i and j; where the keys are hashed in a predecided order. This also provides an additional layer of indirection, using hash functions, for an attacker doing cryptanalysis on captured encrypted data.
3) Or the two keys could simply be concatenated $K_{ij}||K_{ji}$ to increase the key length.

Using commuting matrices increases network resilience as the attacker would need to determine the entire K matrix rather than only half of it as is the case when K was symmetric. Recall that by using commutativity we have eliminated the requirement of K being symmetric. This is in contrast with other methods that either implement Blom's scheme multiple times to increase the number of keys shared (example to share two keys, implement Blom's scheme twice) [12] or Chan et al.'s scheme [3], called q-composite scheme, that shares at least q keys between nodes to increase network resilience by decreasing the key pool size.

The computational complexity, when using commutative matrices, remains linear as it requires multiplication of a row and a column for each key.

VI. CONCLUSIONS

We have proposed two algorithms for key agreement between nodes in a network. The first algorithm factors a symmetric matrix K into two factors X and Y and the second algorithm chooses matrix X and Y such that they commute and Y is symmetric. In the latter method, matrix K need not be symmetric. We have provided several constructive algorithms to choose X and Y in order to decrease the computational load during pre-deployment phase. Key generation only requires the multiplication of a row and a column of a matrix and is linear in complexity. The commutative method provides two keys per node pair that can be used in different ways depending on security requirements and encryption algorithm used.

REFERENCES

[1] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the CCS'02*. New York, NY, USA: ACM, 2002, pp. 41–47.
[2] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *INFOCOM 2004*, vol. 1, March 2004.
[3] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of 2003 Symposium on Security and Privacy*, may 2003, pp. 197–213.

[4] B. Tas and A. Tosun, "Mobile assisted key distribution in wireless sensor networks," in *2011 IEEE International Conference on Communications (ICC)*, June 2011, pp. 1–6.
[5] F. Liu, X. Cheng, L. Ma, and K. Xing, "SBK: A self-configuring framework for bootstrapping keys in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 7, no. 7, pp. 858–868, July 2008.
[6] K. Gagneja, "Pairwise key distribution scheme for two-tier sensor networks," in *2014 International Conference on Computing, Networking and Communications (ICNC)*, Feb 2014, pp. 1081–1085.
[7] M. A. Simplício, Jr., P. S. L. M. Barreto, C. B. Margi, and T. C. M. B. Carvalho, "A survey on key management mechanisms for distributed wireless sensor networks," *Computer Networks*, vol. 54, no. 15, pp. 2591–2612, Oct. 2010.
[8] S. Ruj and K. Sakurai, "Secure and privacy preserving hierarchical wireless sensor networks using hybrid key management technique," in *2013 IEEE Global Communications Conference (GLOBECOM)*, Dec 2013, pp. 402–407.
[9] S. Sanyal, "Spike: A novel session key management protocol with time-varying secure cluster formation in wireless sensor networks," in *2013 Eleventh Annual International Conference on Privacy, Security and Trust (PST)*, July 2013, pp. 151–160.
[10] W. Bechkit, Y. Challal, and A. Bouabdallah, "A new class of hash-chain based key pre-distribution schemes for WSN," *Computer Communications*, vol. 36, no. 3, pp. 243–255, Feb. 2013.
[11] R. Blom, "An optimal class of symmetric key generation systems," in *Proceedings of the EUROCRYPT 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques*. Springer-Verlag New York, Inc., 1985, pp. 335–338.
[12] W. Du, J. Deng, Y. Han, P. Varshney, J. Katz, and A. Khalili, "A pairwise key predistribution scheme for wireless sensor networks," *ACM Transactions on Information and System Security*, vol. 8, pp. 228–258, May 2005.
[13] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-secure key distribution for dynamic conferences," in *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, ser. CRYPTO '92. London, UK: Springer-Verlag, 1993, pp. 471–486.
[14] E. K. Wang, L. C. Hui, and S.M.Yiu, "A new key establishment scheme for wireless sensor networks," *International Journal of Network Security & Its Applications*, vol. 1, no. 2, pp. 17–27, July 2009.
[15] W. Chunying, L. Shundong, and Z. Yiying, "Key management scheme based on secret sharing for wireless sensor network," in *Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on*, Sept 2013, pp. 574–578.
[16] M. Bertier, A. Mostefaoui, and G. Tredan, "Low-cost secret-sharing in sensor networks," in *2010 IEEE 12th International Symposium on High-Assurance Systems Engineering (HASE)*, Nov 2010, pp. 1–9.
[17] K. Gagneja, "Pairwise key distribution scheme for two-tier sensor networks," in *2014 International Conference on Computing, Networking and Communications (ICNC)*, Feb 2014, pp. 1081–1085.

Abhishek Parakh is an Assistant Professor of Information Assurance at the College of Information Science and Technology at University of Nebraska, Omaha. He received his Ph.D. in Computer Science from Oklahoma State University. He is also a member of Nebraska University Center for Information Assurance, a National Security Agency (NSA) designated Center for Academic Excellence in Information Assurance Education - Cyber Defense (CAE-IA/CD). His research interests include cryptographic engineering, distributed systems security, resource constrained encryption systems, protocol development, quantum cryptography and SCADA systems security. He has over 35 publications and has been funded by NSF and NASA.

Subhash Kak is Regents Professor in the School of Electrical and Computer Engineering at Oklahoma State University in Stillwater. Prior to joining Oklahoma State University, he served for many years as the Delaune Distinguished Professor of Electrical and Computer Engineering at Louisiana State University in Baton Rouge.

He is the author of several books that include The Nature of Physical Reality (New York, Peter Lang, 1986) and The Architecture of Knowledge (New Delhi, CRC, 2004). His areas of interest include data security, quantum computing, information theory, neural networks, and history of science. His technical research is in the fields of data security and cryptography, information theory, neural networks, and quantum information and he has also written on archaeoastronomy and art. Amongst his awards are British Council Fellow (1976), Science Academy Medal of the Indian National Science Academy (1977), Kothari Prize (1977), UNESCO Tokten Award (1986), Goyal Prize (1998), National Fellow of the Indian Institute of Advanced Study (2001), and Distinguished Alumnus of IIT Delhi (2002).