KEY MANAGEMENT AND ENCRYPTION IN WIRELESS SENSOR NETWORKS USING HADAMARD TRANSFORMS

By

SINGI REDDY ROHITH REDDY

Bachelor of Technology in Computer Science and Engineering

Jawaharlal Nehru Technological University

Hyderabad, India

2009

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2011

KEY MANAGEMENT AND ENCRYPTION IN WIRELESS SENSOR NETWORKS USING HADAMARD TRANSFORMS

Thesis Approved:

Dr. Subhash Kak

Thesis Adviser

Dr. Johnson Thomas

Dr. David Cline

Dean of the Graduate College

ACKNOWLEDGMENTS

It is my pleasure to acknowledge and gratify all those people who helped me in completing my thesis.

First and foremost, I show my gratitude and immense respect towards my thesis Advisor Dr. Subhash Kak without whom my thesis would not have been successful. It was his who had been a source of inspiration and his advice ignited a strong desire in me to do my thesis on Key Management. He was the one who has always guided and motivated me in getting new ideas and implementing them in practice. His strong support and timely advice has inspired me to further explore this topic. It was him who made my dream of publishing a technical paper come true and I feel no person can have a better advisor than him.

I would like to express my gratitude to Dr. Johnson Thomas and Dr. David Cline for supporting me in further exploring this topic. It was them who offered valuable assistance and helped me to rectify my mistakes.

I would like to express my special gratitude to Dr. Allen Finchum and Bruce Battles for encouraging and supporting me throughout my study at OSU. It was their trust and motivation which made me to develop confidence and could successfully complete the projects which were given to me.

I would like to thank my friends, Venkata Ravinder Paruchuri, Aileni Anvesh Reddy, Yashwanth Kothapalli, Sidharth Eechampati, Pradeep Kumar Dantala, Nitesh Reddy, Chakradhara Reddy, Vijay Singh, Abhishikta Putta, Himabindu Visireddy, Bipul Chandra, Sanath Chilaka, Sagar Kodukula, Swapnika Ratakonda, Anish Kopulla, Raviteja Gunda, Rohit Vaidya, Sujith Reddy

Beemireddi, Shashank Sadalia and Harrikishan Kotha for always supporting me throughout my research.

Lastly, I would like to thank my brother Srikanth Reddy who has always been with me during my hard times and helped me in getting new ideas and made me to gain confidence in completing my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Wireless Sensor Networks

Wireless networks can be categorized according to their application, one of which is sensor networks. Sensor networks have increasingly become the subject of scientific interest over the past few years. Sensor networks comprise a number of sensors with limited resources that are used to collect environmental information. These sensor networks have been considered for various purposes including security monitoring, target tracking and research activities in hazardous environments [31]-[34]. These networks can vary in size and design, depending on the application. The sensors usually communicate with each other via a wireless channel. Since the sensors may be located in a hostile environment, such as in military applications, security is a critical issue. Consider a situation where an adversary could simply access the wireless channel to obtain information, or by physically capturing sensors distribute false information to the network. Therefore, security considerations, such as authentication and confidentiality must be undergone to ensure integrity of node and proper functionality of the network. Since authentication and confidentiality protocols require an agreed key between the entities and secure communication depends on the cryptographic schemes, key management is a very important security issue in wireless sensors networks [1], [29], [30]. In the past, key management protocols were based on either symmetric or asymmetric cryptographic functions. Due to resource limitations and security in the sensors, key management protocols based on public keys are not suitable.

Hence, symmetric algorithm based key management protocols are considered here for use in wireless sensor networks.

Several people have proposed different approaches to address the problem of key management [22]-[27]. The simple and easy method that could be employed is an online key management center via an access point. But, this approach requires high overhead to establish shared keys between sensors [13]. A more interesting method is to pre-distribute keys among the sensors, which entails lower cost. Considering this approach, various schemes have been proposed, but many of them have their own security or efficiency problems. Some of these protocols try to increase the efficiency of the key management by using deployment knowledge in the pre-deployment phase and to classify the sensors into groups for the distribution of secret information. To increase connectivity and reduce the overhead between sensors, several people proposed different ideas in grouping the nodes in order to easily establish a common key. In particular, key agreement schemes for wireless sensor networks must satisfy the following requirements.

(1) Low energy consumption: Since sensor nodes consist of batteries which have limited power, the scheme which is employed for key agreement should have low communication and computation overhead.

(2) Low cost: Since many numbers of sensors are used for effectively constructing a network, a cost effective scheme should be employed.

(3) Low memory usage: Since sensor nodes are manufactured with limited memory, the memory requirements of the scheme should be low.

(4) Lack of trusted infrastructure: Since sensor nodes are usually scattered over a wide area which is unattended and may be in hostile environment, no node should be trusted as there is possibility for a node to get captured.

(5) Resilient against node capture: The resilience of the scheme should be high, where resilience refers to the percentage of communication links not involving compromised nodes which remain secure even if a group of nodes are compromised. A scheme is perfectly resilient if the compromise of any node does not compromise the security of any communication channels between non-compromised nodes.

To develop a high performance key management protocol, factors such as processing overhead, resource consumption, connectivity and resilience against compromised sensors should be considered. But, some of these factors are contradictory i.e., increasing network connectivity increases sensor memory requirements. Hence, some approaches having proper connectivity use large part of memory and processing power while others using less memory have low connectivity.

It can be observed that most of the schemes have concentrated on one or more factors and none of them proved to be perfect in efficiently establishing the key between nodes. The main problem in most of the key management schemes is managing the memory, since sensor nodes are provided with limited resources there should be appropriate data selected to store in the sensor node. In this thesis, our objective is to model a scheme which reduces the computation cost and memory cost to certain extent. Since key establishment is not sufficient for two nodes to communicate, there should be an encryption scheme applied in order to maintain the confidentiality of data. To ensure security a new encryption scheme is proposed and all the operations are based on key generated in key management scheme.

Key establishment between any pair of nodes is an essential requirement for providing secure services in wireless sensor networks. Blom's scheme [3] is a prominent key management scheme but its shortcomings include large computation overhead and memory cost. In this thesis we propose a new scheme that modifies Blom's scheme [3] which in turn reduces memory and

computation costs. The proposed key management scheme is based on the pre-distribution scheme of Blom [3] and encryption scheme is based on the Hadamard transform [2] which is an example of generalized class of Fourier transforms. Hadamard transform can be defined either recursively or by using binary representation and its symmetric form lends itself to applications ranging across many technical fields such as data encryption [4], signal processing [5], data compression algorithms [6], randomness measures [7], and so on.

In the proposed scheme original public matrix (Vandermonde matrix [17]) used in Blom's scheme [3] is replaced with non-binary Hadamard matrix. This modification helps to reduce the complex computations involved in the Vandermonde matrix [17] and also avoids the overhead of storing the columns of pubic matrix in the sensor memory. The new scheme also gives the value for secure parameter t such that the network is resilient within the limit. Thus, the simple modification lead to decreasing computation, communication and memory cost. The encryption scheme proposed in this thesis is a new technique which makes use of the non-binary Hadamard matrix.

The rest of the document is organized as follows. Chapter 2 describes the previous related work done on key management and encryption. Chapter 3 describes the proposed scheme with an enhancement to the Blom's scheme [3] and new encryption technique. Chapter 4 describes implementation, project management and future work to be done.

CHAPTER II

REVIEW OF LITERATURE

Key Management Scheme's

Recent advances in both electronic and computer technologies have widened the way for the increasing demand of wireless sensor networks. Key management is the critical issue in wireless sensor networks. Several people have proposed different approaches to address the problem of key management.

Recently, two random key distribution schemes suited for sensor networks were proposed. The first and foremost is random key pre-distribution scheme for wireless sensor networks proposed by Eschenauer and Gligor [8]. In this scheme before deployment of the sensors, some keys from the large key pool are selected and stored randomly in the sensor memory. Upon deployment, sensors look for a common key to establish secure communication. If a node doesn't find any, a common key can be established via an intermediate node which has common keys with both the nodes. In this scheme, the size of the pool is an important factor for connectivity and security. Here, network connectivity is defined as the probability of direct key generation between any two neighboring nodes. As size of the key pool increases, connectivity deceases by increasing the security. Since the scheme uses the method of randomly distributing the keys, it may be quite possible for a pair of sensor nodes not to establish a common key.

A generalization of the Eschenauer and Gligor [8] is the q-composite scheme proposed by Chan Perrig, and Song [9] which improves the resilience of the network for the same amount of key storage and requires an attacker to get many more nodes in order to compromise on additional communication links. The difference between this scheme and the previous one is that the q-composite scheme requires two nodes to find q (with $q > 1$) keys in common before deriving a shared key and establishing a secure communication link. It is shown that, by increasing the value of q, network resilience against node capture is improved for certain ranges of other parameters. Their scheme achieves good security under small scale attacks but increases vulnerability in case of a large scale node compromise attack.

Du et al. [10] proposed a key management protocol based on Eschenauer and Gligor [8], which uses deployment knowledge during key distribution. In this protocol, deployment knowledge is modeled using a Gaussian probability distribution function (pdf). On the other hand, the methods with no deployment knowledge use uniform distribution, enabling the sensors to reside anywhere in the network with equal probability. In this method the network area is divided into square cells and each cell is associated with one group of sensors. Key-pool is divided into sub-key-pools for each cell such that each sub-key-pool has some key correlation with its neighboring sub-key-pools. G(i, j) is defined as the sub-key-pool of cell (i, j). Each sensor in a cell, randomly selects m keys from its associated sub-key-pool. Therefore, using deployment knowledge allows the selection of random keys from a smaller sub-key-pool, which improves the protocol performance especially in large scale networks. In addition, if some sensors are compromised, then other sensors have greater security than those with the approach in [8]. Even this method has some problems in producing common keys between nodes.

Zhou et al. [11] proposed another key management protocol called LAKE which is based on a symmetric polynomial and deployment knowledge. Similarly in this scheme, the network is

divided into square cells and each cell is associated with a group of sensors. The polynomial in this method is a t-degree tri-variate symmetric polynomial. Each sensor in this protocol has credentials (n1, n2), where n1 represents the cell identity and n2 represents the sensor identity. Based on these credentials, a polynomial share is calculated for each sensor and stored on it. After deployment, sensors that have one mismatch between their credentials can directly generate a common key. So in this scheme, all sensors belonging to a cell can establish common key with each other directly. However, only two specific sensors belonging to different cells can establish common keys with each other directly.

Lin et al. [12] proposed another key management protocol called LPBK in which the network area is also divided into square cells. Each cell has a specific symmetric polynomial which is used to generate a polynomial share for the sensors in the corresponding cell and the four adjacent (vertical and horizontal) cells. In this scheme, a sensor must store five polynomial shares in its memory. Therefore, every sensor can then directly generate a common key with the other sensors in the same and four adjacent cells.

Blom [3] proposed a key distribution method that allows any pair of users in a system to be able to find a unique shared key. This method says, a network with N users and a collusion of less than t+1 users cannot reveal the keys which are held by other users, which means the security of the network depends on the chosen value of t and t here is called as Blom's secure parameter, where t<<N. Larger value of t leads to greater resilience but should be cautious while choosing the value because a very high value of t increases the amount of memory required to store key information.

During the initialization phase, a central authority or base station first constructs a (t + 1) x N matrix P over a finite field GF(q), where N is the size of the network and q is the prime number. P is known to all users and it can be constructed using Vandermonde matrix, it can be shown that any t+1 columns of P are linearly independent when n_i, i=1, 2,…N are all distinct.

$$P = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ n_1 & n_2 & n_3 & \cdots & n_N \\ {n_1}^2 & {n_2}^2 & {n_3}^2 & \cdots & {n_N}^2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ {n_1}^t & {n_2}^t & {n_3}^t & \cdots & {n_N}^t \end{bmatrix}$$

Then the central authority selects a random (t + 1) x (t + 1) symmetric matrix S over GF(q), where S is secret and only known by the central authority. An N x (t + 1) matrix $A = (S. P)^T$ is computed. Because S is symmetric, it is easy to see

$K = A.P = (S. P)^T. P = P^T. S^T.P$

$= P^T.S.P = (A.P)^T = K^T$

User pair (i, j) will use K_{ij}, the element in row i and column j in K, as the shared key. Because K_{ij} is calculated by the i-th row of A and the j-th column of P, the central authority assign the i-th row of A and the i-th column of P to each user i, for 1, 2….. N. Therefore, when user i and user j need to establish a shared key between them, they first exchange their columns of P, and then they can compute K_{ij} and K_{ji}, respectively, using their private rows of A. The t-secure parameter guarantees that no coalition of up to t nodes has any information about K_{ij} or K_{ji}.

Yu et al. [13] proposed another key management protocol based on the Blom's scheme [3]. Here, the network area is divided into hexagonal cells and the information from the associated Blom matrices is stored in the sensors, based on deployment knowledge. A secret Blom's matrix A_i (i=1, 2...N) is allocated to each cell, and each sensor is provided with a row from the matrix. Hence, the sensors belonging to a cell can produce a common key directly. To generate a common key between sensors in different cells, another set of matrices (B) is constructed. To allocate these matrices, two parameters b and w are defined where b is the number of B matrices allocated to a group and w is the number of rows selected by each sensor from this group.

The analysis in this scheme shows that the best results are obtained when w = 2 and b = 2. Initially, the cells are divided into base cells and normal cells where base cells are not neighbors, but normal cells are neighbors with two base cells. To produce a common key between sensors in

neighboring cells, a matrix B_i is allocated to each base cell together with its six neighbors. The Blom's scheme is used on this matrix to produce the required information for the sensors. Since each normal cell is a neighbor with two base cells, their sensors receive information from two matrices B_i and B_j. Though it proves to have good connectivity, but the memory utilized is high and depends on structure of the grid. Dividing the cells in hexagon proved that, the number of neighbors effected when a node is compromised is less when compared to square and triangle [14] grid methods.

	Triangle	**Square**	**Hexagon**
Neighbor groups	12	8	6
Affected groups	31	21	13
Rows stored	4	4	3

Table 1. Number of Affected Groups for Different Grid Shapes [13]

Blundo et al. [15] proposed several schemes where any group of n nodes can generate a common key. Blundo's scheme [15] uses n variables polynomials with t-degree to establish key distribution between the nodes with t-secure property. Comparing to Blom's scheme instead of using matrices this scheme generates a bi-variate polynomial and this polynomial is used in generating the keys between nodes. The basic problem with this scheme is the inability to directly apply to the sensor networks due to memory overhead for storing keys. Later, the work on this was based on using pre deployment knowledge [16] which proved to be advantageous over the original Blundo's scheme.

A different method of key management in sensor networks is given in [21]. This method requires marking the sensors in an ordered manner. Data in distributed networks can also be protected by using the method of partitioning [18]-[20].

Hadamard Transform

Security of data is a critical issue not only in wireless sensor networks but in any network. Though keys are necessary for establishing a communication link between nodes but for secure

communication, information should be encrypted before exchanging between any parties. There are several encryption schemes proposed in this regard. All the computations in the proposed encryption scheme are based on non-binary Hadamard matrix.

CHAPTER III

METHODOLOGY

We can observe that most of the proposed key management schemes were based on Blom's scheme. Though, Blom's scheme [3] was successful in computing the key between any two nodes, but this scheme utilized more memory to store the keys for strong connectivity and resilience against node capture. The original Blom's scheme can be modified in such a way that all the other schemes which rely on this can reduce the memory and computation overhead to certain extent.

Modified Key Management Scheme

The proposed method makes use of the original Blom's scheme [3]. In the original Blom's scheme all the computations involved in generating the keys were based on Vandermonde matrix which is a public matrix (P) and known to even the adversaries. Here, to make sure that any t+1 columns of P are linearly independent i.e., to make sure that unique keys are generated, all the values in matrix are chosen to be distinct. However, for a large value of t, number of rows in the matrix increase and this in turn correspond to a greater value in the columns, this is because the column values in the public matrix increases in a geometric series. In Blom's scheme [3] for any two nodes to generate a common key, each node should store column of public matrix and row of the calculated secret matrix. Since every sensor node is provided with limited resources, it is difficult to store both the row and column values in sensor memory.

To reduce computation and memory overhead in Blom's scheme, instead of using Vandermonde matrix [17] a non binary Hadamard matrix can be used as the public matrix. The results of this enhancement are proved to be more effective than the original Blom's scheme. As the Hadamard matrix is a square matrix with 1s and -1s, it reduces the complexity of calculating values for all the elements corresponding to the columns in Vandermonde matrix. Since every node can easily generate a Hadamard matrix of known size, the cost of storing columns in the sensor memory can be reduced. Similar to Blom's scheme all the operations which are to be performed in the enhanced method depends on the prime number i.e., a number which gives the desired key length.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

Figure 1. 4x4 Hadamard Matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 30 & 1 & 30 \\ 1 & 1 & 30 & 30 \\ 1 & 30 & 30 & 1 \end{bmatrix}$$

Figure 2. Non-Binary Hadamard Matrix

Binary form of Hadamard matrix is depicted in Figure 1. Only change that is made to the Hadamard matrix is, all negative numbers are replaced with a non-negative prime number. We can observe that Hadamard matrix shown in Figure 2 has equal numbers of one's and prime number, this simplifies the further calculations. Key generation technique in modified scheme is similar to original Blom's scheme [3]. Following are the steps involved in calculating the key.

(1) Initially N x N form of Hadamard matrix is considered and depending on the t value first t rows along with N columns are selected as a public matrix. The construction of Hadamard matrix depends on the prime number (q), we must set q to be larger than the network size (q>N).

(2) The central authority selects a random (t + 1) x (t + 1) symmetric matrix S, where S is secret and only known to the central authority. An N x (t + 1) matrix $A = (S. P)^T$ is constructed.

(3) The central authority stores each row of matrix A in the node memory with corresponding index. This is shown in Figure 3.

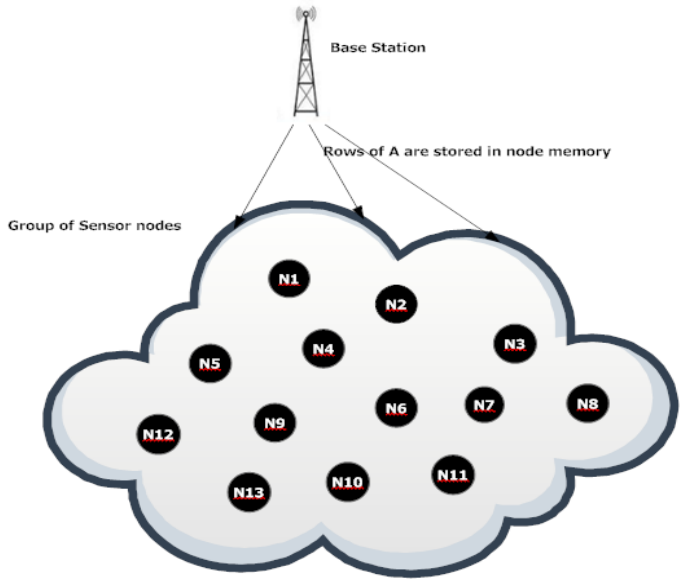

Figure 3. Storing Rows into Node Memory

(4) Finally user pair (i, j) can compute the key by generating the Hadamard matrix and multiplying the secret row stored in the node with column of the Hadamard matrix corresponding to the node index with which it want to generate the key. The key generation between any two nodes is shown in Figure 4.

By integrating the proposed scheme along with the schemes which previously relied on Blom's scheme such as Du. et al. [10] we can gain significant memory utilization along with reduced computations.

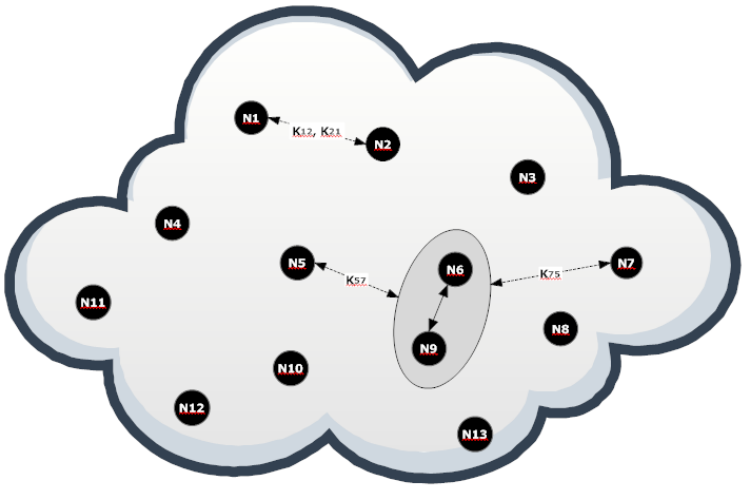

Figure 4. Key Establishment between any Two Nodes

Example for Key Management Scheme

The following example shows the working of modified Blom's scheme using Hadamard matrix. Let, the number of nodes in the network be 8, secure parameter t=6 and prime number q=31 which says if no more than 6 nodes in the network are compromised it is not possible to find the keys of other users. An assumption is made such that all the sensor nodes are within the transmission range and can communicate directly.

Modified Hadamard Matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 30 & 1 & 30 & 1 & 30 & 1 & 30 \\ 1 & 1 & 30 & 30 & 1 & 1 & 30 & 30 \\ 1 & 30 & 30 & 1 & 1 & 30 & 30 & 1 \\ 1 & 1 & 1 & 1 & 30 & 30 & 30 & 30 \\ 1 & 30 & 1 & 30 & 30 & 1 & 30 & 1 \\ 1 & 1 & 30 & 30 & 30 & 30 & 1 & 1 \\ 1 & 30 & 30 & 1 & 30 & 1 & 1 & 30 \end{bmatrix}$$

Public Matrix (P)

$$P=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 30 & 1 & 30 & 1 & 30 & 1 & 30 \\ 1 & 1 & 30 & 30 & 1 & 1 & 30 & 30 \\ 1 & 30 & 30 & 1 & 1 & 30 & 30 & 1 \\ 1 & 1 & 1 & 1 & 30 & 30 & 30 & 30 \\ 1 & 30 & 1 & 30 & 30 & 1 & 30 & 1 \end{bmatrix}$$

Let secret matrix (S) is any symmetric matrix.

$$S=\begin{bmatrix} 3 & 11 & 15 & 28 & 7 & 5 \\ 11 & 30 & 4 & 1 & 2 & 8 \\ 15 & 4 & 6 & 14 & 18 & 21 \\ 28 & 1 & 14 & 17 & 25 & 6 \\ 7 & 2 & 18 & 25 & 27 & 9 \\ 5 & 8 & 21 & 6 & 9 & 8 \end{bmatrix}$$

$A= (S.P)^T$

$$S.P=\begin{bmatrix} 69 & 1345 & 1316 & 968 & 417 & 1403 & 1664 & 1026 \\ 56 & 1187 & 201 & 1274 & 346 & 1013 & 491 & 1100 \\ 78 & 1209 & 658 & 977 & 1209 & 1122 & 1789 & 890 \\ 91 & 787 & 990 & 700 & 990 & 1338 & 1889 & 1251 \\ 88 & 1132 & 1335 & 929 & 1132 & 1654 & 2379 & 1451 \\ 57 & 695 & 840 & 1130 & 550 & 724 & 1333 & 1159 \end{bmatrix} \text{ mod } 31$$

$$A= (S.P)^T = \begin{bmatrix} 7 & 25 & 16 & 29 & 26 & 26 \\ 12 & 9 & 0 & 12 & 16 & 13 \\ 14 & 15 & 7 & 29 & 2 & 3 \\ 7 & 3 & 16 & 18 & 30 & 14 \\ 14 & 5 & 0 & 29 & 16 & 23 \\ 8 & 21 & 6 & 5 & 11 & 11 \\ 21 & 26 & 22 & 29 & 23 & 0 \\ 3 & 15 & 22 & 11 & 25 & 12 \end{bmatrix}$$

After calculating matrix A each sensor node memory is filled with unique row chosen from matrix A corresponding to the node index. It shows that all the computations involved in calculating the matrix A are simple.

Generating Key

Any pair of nodes must generate a key for securely communicating with each other. Since Hadamard matrix is easy to construct, the overhead of storing a column in node memory is reduced. Suppose node two and node eight want to communicate with each other, first they generate the column of neighboring node using the Hadamard matrix and then multiply it with the row stored in node memory.

$$K_{2,8}=A_2.P_8 = \begin{bmatrix} 12 & 9 & 0 & 12 & 16 & 13 \end{bmatrix} \begin{bmatrix} 1 \\ 30 \\ 30 \\ 1 \\ 30 \\ 1 \end{bmatrix} = 787 \text{ mod } 31 = 12$$

$K_{8,2}=A_8.P_2=\begin{bmatrix}3 & 15 & 22 & 11 & 25 & 12\end{bmatrix}\begin{bmatrix}1\\30\\1\\30\\1\\30\end{bmatrix}=1190 \bmod 31=12$

We can observe that both nodes generate a common key and for further communication they make use of the pair-wise key.

In general matrix K can be represented as shown below and we can notice that the symmetric nature gives the same key for any pair of nodes such that $K_{ij} = K_{ji}$.

K= A.P

$$K=\begin{bmatrix}5 & 0 & 8 & 26 & 25 & 0 & 28 & 26\\0 & 25 & 7 & 18 & 4 & 19 & 11 & 12\\8 & 7 & 29 & 20 & 29 & 9 & 19 & 22\\26 & 18 & 20 & 22 & 0 & 17 & 25 & 21\\25 & 4 & 29 & 0 & 9 & 18 & 13 & 14\\0 & 19 & 9 & 17 & 18 & 19 & 27 & 17\\28 & 11 & 19 & 25 & 13 & 27 & 4 & 10\\26 & 12 & 22 & 21 & 14 & 17 & 10 & 26\end{bmatrix}$$

The proposed scheme presents an alternative way of generating the keys using Blom's scheme. This scheme has an advantage of reducing computation overhead and memory costs over the original Blom's scheme and also gives a minimum value for the secure parameter t such that the network is more resilient. It is observed that several decomposition schemes such as LU, LQ, QR, etc can be used to generate a key between any pair of nodes.

Encryption Scheme

In wireless sensor networks, apart from key management it is essential to provide security to the data. A chained encryption scheme is proposed which makes use of the key generated in the previous step. Initially before applying the scheme on data, each digit in the key should be scaled to nearest prime number or the scaling can even be decided by the nodes which are communicating. Therefore, the security level or depth of encryption depends on the number of digits in the key. Similar to key management, encryption also uses non-binary Hadamard

matrices. To do so, Hadamard transform is generalized such that all the values in the matrix are non negative. Each negative number is replaced with corresponding modulo number; for example while performing modulo 7 operations -1 is replaced with 6 to make the matrix non-binary. Only prime modulo operations are performed because non prime numbers can be divisible with numbers other than 1 and itself. The use of modulo operation along with generalized Hadamard matrix limits the output range and decreases computation complexity.

In the proposed method, the input sequence is split into certain group of bits such that each group bit count is a prime number. As we are dealing with non-binary numbers, for each group of binary bits their corresponding decimal value is calculated. The input sequence is split in such a way that, the maximum decimal value thus obtained by the group is a prime number. The reason to consider only prime number is because our operations are on modulo prime numbers after converting into the non-binary form.

Both during encryption and decryption the sequence after converting into the non-binary form is broken down into chunks of equal size as a power of 2. Moreover, the whole input sequence is divided into groups of equal size and each group is then multiplied with the corresponding Hadamard matrix. The Hadamard matrix which is to be multiplied is chosen based on the maximum value of the decimal sequence. For example in a sequence of groping of three bits, the maximum value that can be obtained is 2^3-1=7, and hence Hadamard matrix of modulo 7 is used to perform the operations corresponding to the sequence.

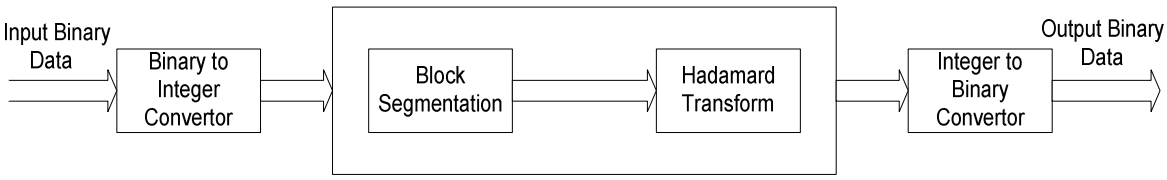

Figure 5. The Encryption System

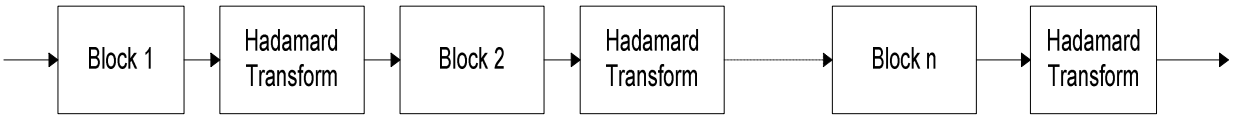

Figure 6. Chaining in Block Sequence which expands the middle block of Figure5

The general scheme of the proposed encryption system is shown in Figure 5 and Figure 6. The power of this is a consequence of the many different ways and the number of times the chaining of blocks is done. Similar pattern can be generated for decryption in the reverse order.

Initially before performing any operations, all the numbers in the decimal sequence need to be analyzed. As, modulo number of the matrix is equal to the maximum number obtained by grouping the sequence, there is a possibility for the sequence to have the maximum values, for example 111 produce 7 which is equal to modulo number of the matrix. To overcome this problem a two dimensional integer array is used. Before performing any operations, at each level the array is filled with 0s or 1s in such a way that, for each maximum value in the decimal sequence 1 is entered in the array corresponding to the index of the maximum value and remaining other indexes are filled with 0s. This scheme of representation helps to keep track of all the maximum numbers in the sequence, if any. And during decryption whenever 0 is encountered in the sequence the corresponding position is checked in the array, if the position shows 1 in the array then 0 in the sequence is replaced with maximum value for that particular sequence.

Data Encryption

The idea of encryption is to multiply the decimated input sequence with the non-binary Hadamard matrix in a chained manner. The encryption is carried out at number of levels which makes it hard to break the encrypted message. Here, the number of levels in which the encryption is carried will depend on the key sequence provided, for example key sequence with number {3 5 7} will perform encryption at three levels one for each number in the key. Moreover, the given key

sequence should contain only prime numbers such that 2^x-1 is also a prime where x is any number in the sequence. This sequence can be considered as the public key based on which both encryption and decryption are carried.

Initially, the first number from the sequence is considered and whole input sequence is grouped based on the number. The decimal values for each group obtained thereafter are calculated. As we are dealing with Hadamard matrix which is a square matrix, the decimated input sequence is divided into chunks of equal size corresponding to the size of the matrix. Once after dividing the decimal sequence and if the size of the sequence is less than size of the matrix then zeros must be appended at the end of the decimal sequence to make it equal to the matrix size. Appending zeros is done at each level for those sequences whose length is less than the matrix size.

After getting the decimal sequence for the first number in the key, the decimal sequence obtained is then represented as a column matrix. Next, the Hadamard matrix is multiplied with the decimal sequence column matrix. The matrix which is to be multiplied by the input sequence depends on the number in the given key. For example, if the number in key is 3 then the matrix representing modulo 7 must be used to perform the multiplication. Therefore, at each level different combinations of the Hadamard matrix is used to perform encryption. Since, the multiplication can generate large values, so to limit the output and to make sure that the original sequence is obtained after decryption, modulo operation is applied on the resultant matrix.

The resultant matrix obtained by performing modulo operation is then converted to the binary sequence and this specifies the end of Level 1. Each level uses the binary sequence obtained from its previous level and follows the same encryption procedure. This process is continued until all the numbers in the key are processed and the binary sequence thus obtained will be the final encrypted message. The final encrypted message may have a length which is equal or different from the input sequence.

Encryption Algorithm

Step1: Firstly, consider the given binary input sequence and the key. The given key can have n numbers such that each number say x in n is a prime and 2^x-1 is also a prime. Also, consider a two dimensional integer array. Here, number of rows is equal to number of elements in the array and number of columns is equal to number of input values at each row.

Step2: Next, consider the first element in the key and group the bits in the input sequence based on the number.

Step3: Now convert each group to corresponding decimal number.

Step4: Depending on the length of input sequence, divide the decimated sequence to equal length sub-sequences such that each sub-sequence length should be expressed as power of 2.

Step5: If length of the sub-sequence cannot be expressed as power of 2, append 0^s at last to make the length equal to other sub sequences length.

Step6: For every sub-sequence, if a number in the sub-sequence is equal to 2^x-1, then corresponding index in the array is marked with 1 and remaining all indexes are marked with 0^s.

Step7: Represent each sub-sequence as a column matrix. Multiply each sub-sequence matrix with the modified Hadamard matrix, such that the matrix of the form modulo 2^x-1 must be used to perform multiplication.

Step 8: Apply modulo 2^x-1 on the resultant values obtained after multiplication.

Step9: Convert each decimal number in the sequence to corresponding binary values.

Step10: Now, consider next element in the key and group the bits of the sequence obtained from the previous step based on the number.

Step11: Repeat Step4 to Step9. The sequence obtained after processing the last element in the key is the final encrypted message.

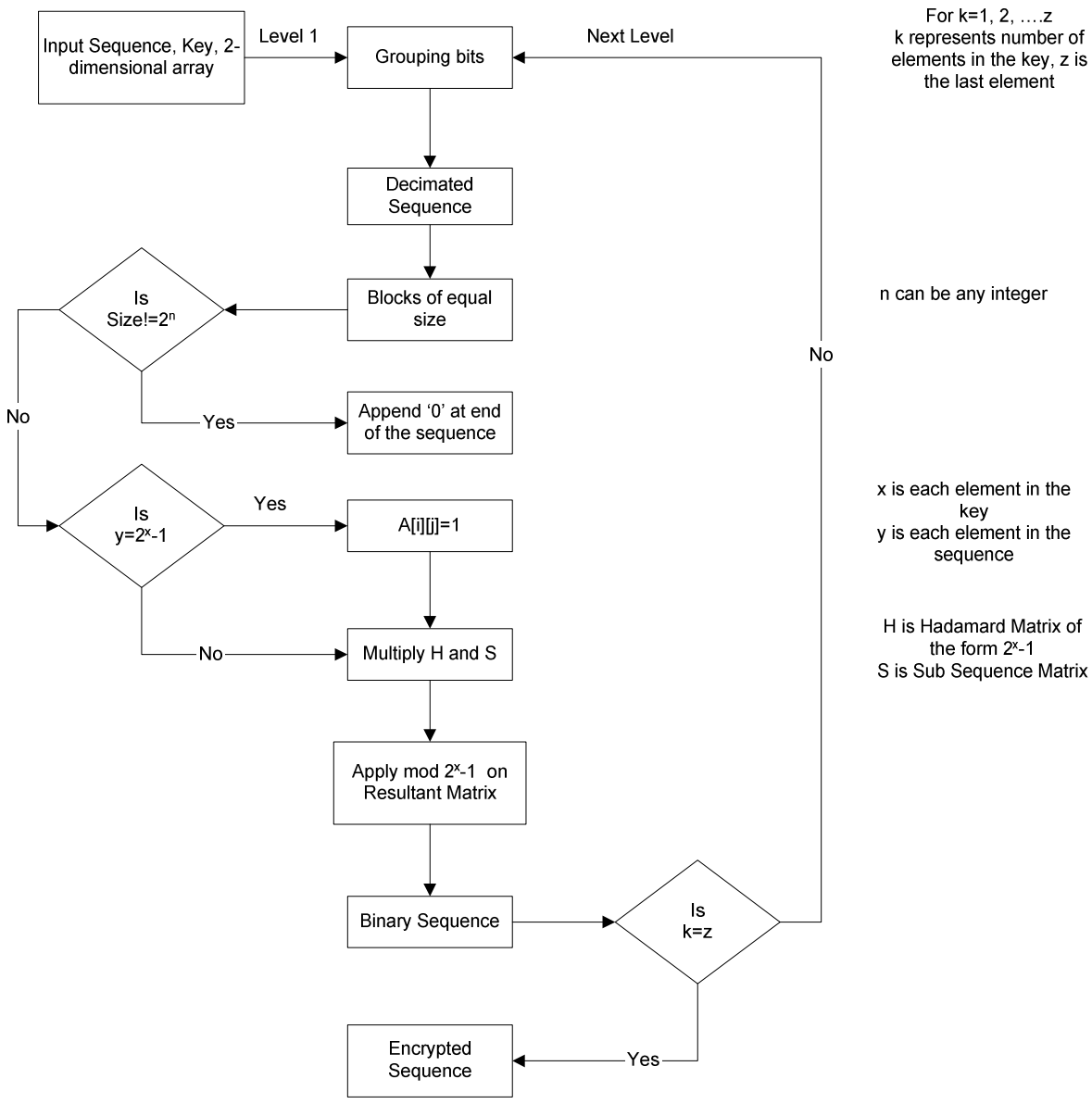

Figure 7. Block Diagram for Encryption

Note: In case if any bits are padded at Level 1 the count for the bits must be noted and after finishing the final level in decryption the last bits of the sequence must be discarded based on the count noted in Level 1 of encryption.

Data Decryption

The decryption scheme discussed in this paper uses the same Hadamard matrix as in encryption with some additional operations. In order to get back the original sequence the decryption should

be performed in a reversed order. Hence, the key sequence is reversed and decryption is performed at levels on each element in the key. For example key of sequence {3 5 7} is reversed as {7 5 3} and similar to encryption each element, in the key is considered to perform decryption. The decryption method is started by considering the first element in the key and then the whole encrypted sequence is grouped based on the number selected. Now, decimated values for each group is calculated and later the sequence is divided into equal sized sub-sequences as done in encryption. Once after division is done, each sub-sequence is then represented by column matrix. In order to perform multiplication, the Hadamard matrix corresponding to modulo 2^x-1 is chosen. For example if grouping is done on element 3 then Hadamard matrix corresponding to 2^3-1 i.e., modulo 7 must be chosen.

After getting the decimated sequence each column matrix is multiplied with the Hadamard matrix, and to get the original message the column matrix must be multiplied with the inverse Hadamard matrix. The inverse Hadamard matrix is same as the original Hadamard matrix but with an additional multiplier operation. To perform decryption using the next element in the key, the decimated output is represented in the binary form.

During encryption we append extra 0s at end of the sequence, so these 0s should be discarded to get the original sequence. Here, only those 0s at the end of sequence which are consecutive must be discarded and make sure that the modified sequence thus obtained should be represented as power of 2. While performing encryption, all the maximum values in the sequence are stored in the array, so here in decryption at each level, for every occurrence of 0 in the sequence, corresponding index is checked in the array. If the index has a value 1 then 0 in the sequence is replaced by the maximum value for that sequence.

Each level thereafter uses the output obtained from the previous level and performs similar operations on it. Thus, the binary sequence obtained after processing the final element in the key will be the original message.

Decryption Algorithm

Step1: Initially, consider the encrypted sequence, reversed key sequence and the array used in encryption.

Step2: Next, consider the first element in the key and group the bits in the encrypted sequence based on the number.

Step3: Now convert each group to corresponding decimal number.

Step4: Depending on the length of input sequence, divide the decimated sequence to equal length sub-sequences such that each sub-sequence length should be expressed as power of 2.

Step5: Represent each sub-sequence as a column matrix. Now, multiply each sub-sequence matrix with the modified Hadamard matrix, such that the matrix of the form modulo 2^x-1 must be used to perform multiplication.

Step6: Now, multiply two modulo 2^x-1 matrices and find the divisor such that the resultant matrix obtained is thus represented as an Identity matrix.

Step7: Calculate modulo multiplicative inverse for the divisor that is, a*y mod 2^x-1 =1[5] where y is the divisor and a is modulo multiplicative inverse.

Step8: Multiply the resultant matrix obtained in Step7 with a.

Step9: Apply modulo 2^x-1 on the resultant values obtained after multiplication.

Step10: For every 0 in the decimal sequence check for the corresponding array index, if the index has an element 1 then replace 0 with 2^x-1.

Step11: Convert each decimal number in the sequence to corresponding binary values.

Step12: Remove all consecutive 0s at end of the sequence such that, the resultant sequence has a length equal to power of 2.

Step13: Now, consider next element in the key and group the bits of the sequence obtained from the previous step based on the number.

Step14: Repeat Step4 to Step12. The sequence obtained after processing the last element in the key is the original message.

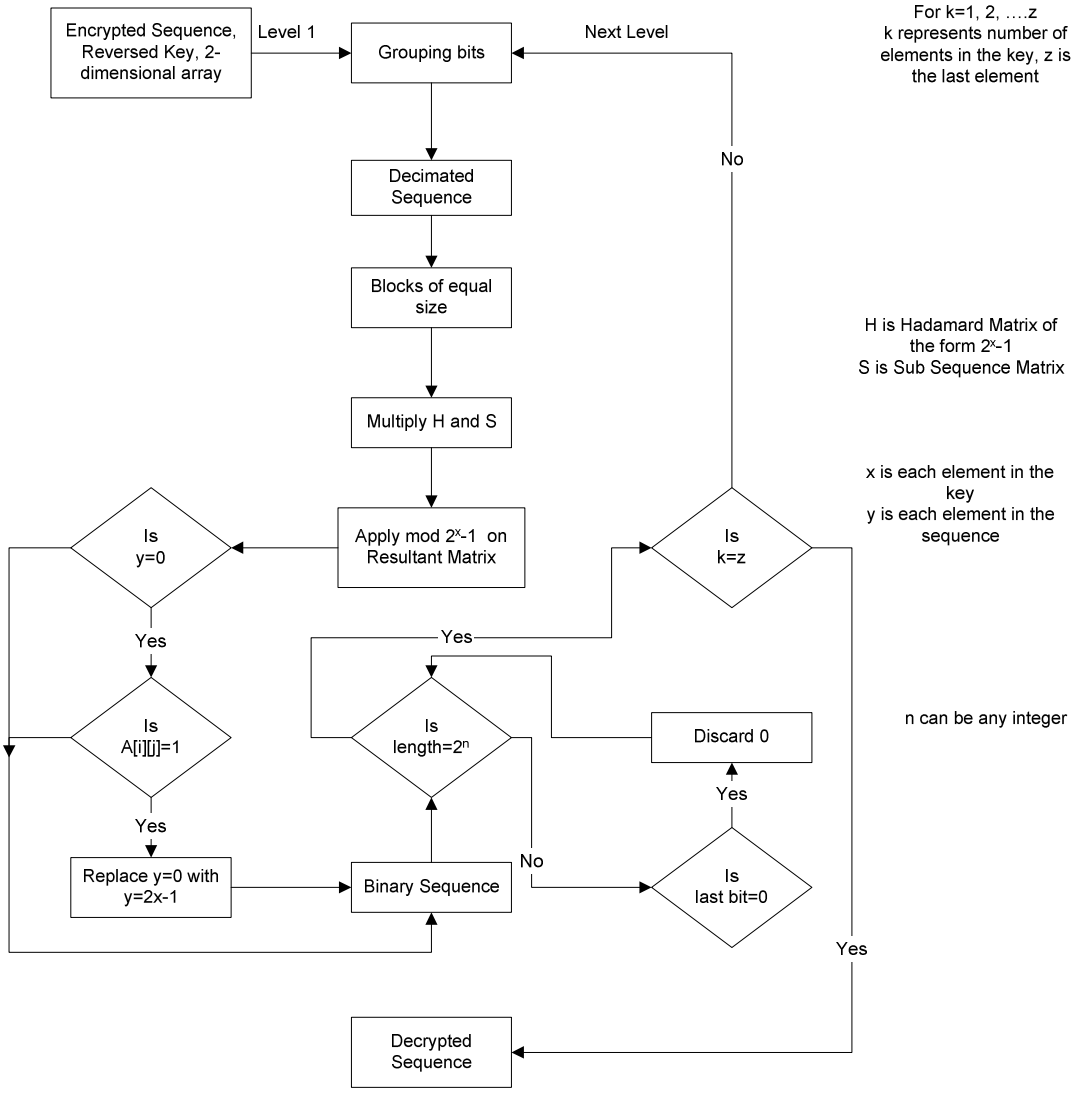

Figure 8. Block Diagram for Decryption

If the input sequence contains all 1s or 0s then the corresponding encrypted sequence is observed to contain all 0s which implies the input does not carry any useful information.

In the proposed scheme structure, the key plays a vital role. It will be challenging for the eavesdropper to break the sequence if the length of the key is large enough and contain distinct elements. On the other hand if the key length is short or if it has many duplicate elements then it can be easily broken using brute force or other techniques.

Thus, both the key management scheme and encryption scheme can be combined to effectively generate the keys and provide security to the data while any pairs of nodes are communicating with each other.

CHAPTER IV

FINDINGS

Modified Key Management using Blom's Scheme

The original Blom's scheme provides a rough estimation for the value of secure parameter t and it do not give a desired range for the parameter such that the entire network is secure within the range. The proposed scheme opens a way to find the value of secure parameter t so that all the nodes in the network can communicate securely.

The Secure Parameter (t)

Simulations are carried on with a network size of 16, 32 and 64 with varying key length. Each network is tested with different value of the parameter t. It is observed that for any network we can generate large number of unique keys if the parameter size chosen is greater than half the size of the network. In general for a network with size N, the value of t should be, t>=N/2+1. It is also observed that the prime number chosen to compute the keys have a great impact on number of unique keys generated in the network. For a minimum prime number it is observed to have maximum number of unique keys at N/2+1 but total number of unique keys increases if the value of prime number is increased.

Figure 4 shows value of t versus number of unique keys for a network with 32 nodes and for prime number 751. It was observed that for a t value greater the half the size of network (t=17) total number of unique keys generated by the network will increase drastically.

Moreover, not much difference was observed if the t value is increased further. Figure 5 shows value of t versus number of unique keys for a network with 64 nodes and for a prime number 991. Similar observation was made and found that total number of unique keys increases drastically at t=33. The total number of unique keys varies for network with varying sizes and different prime numbers.

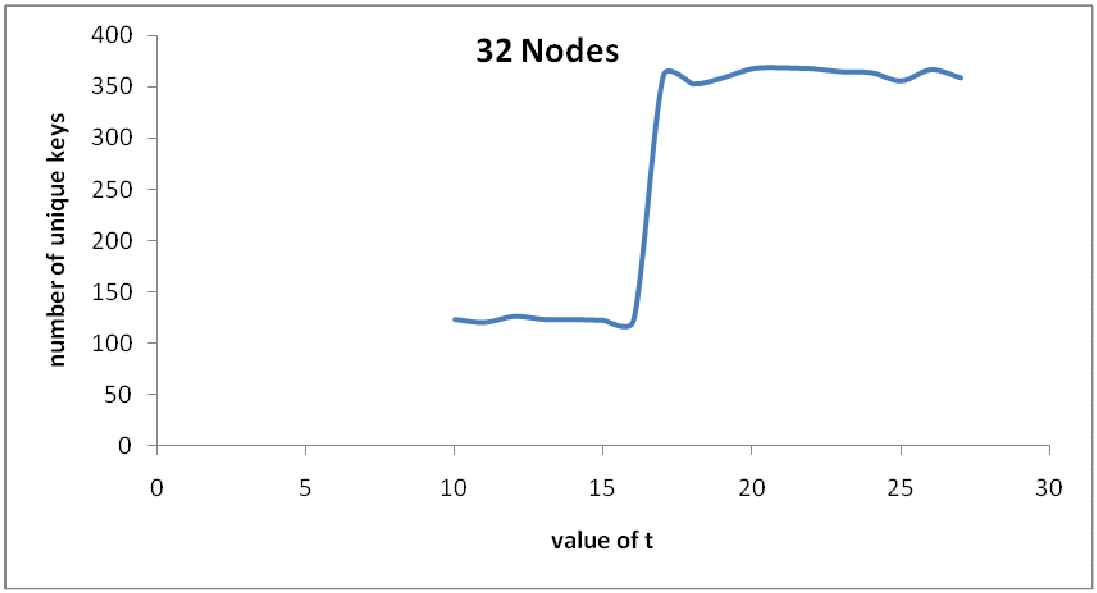

Figure 9. Number of unique keys for a network of 32 nodes

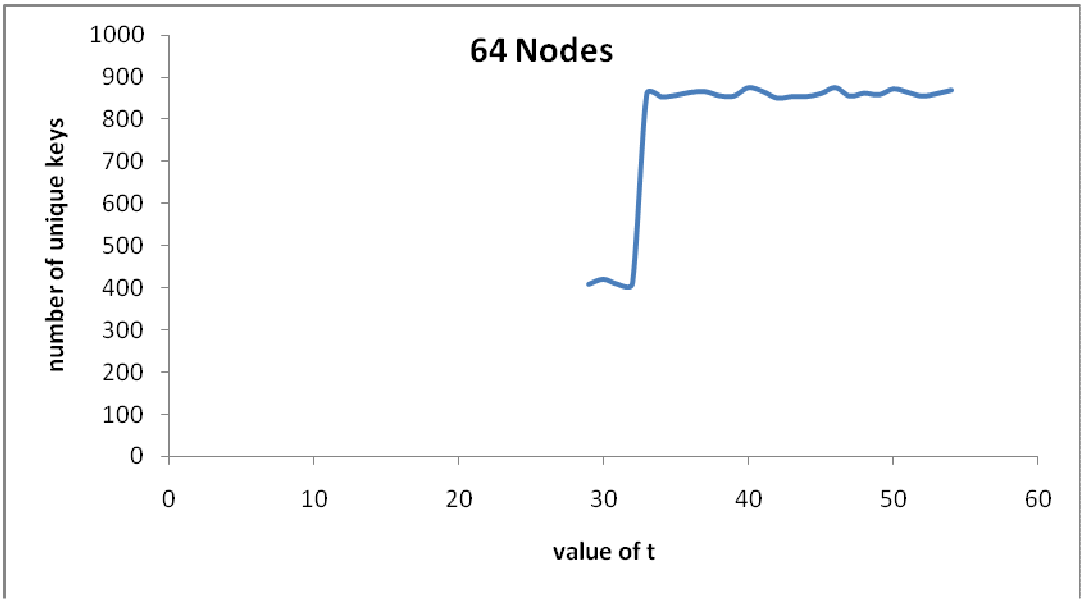

Figure 10. Number of unique keys for a network of 64 nodes

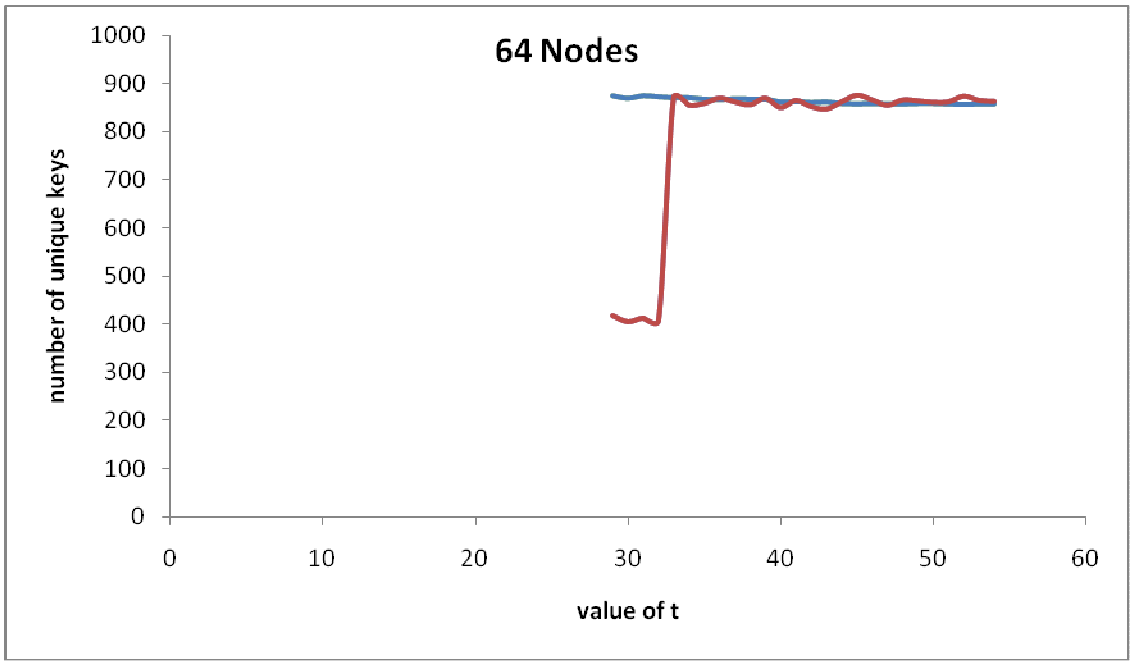

Figure 11. Difference between keys produced in original Blom's scheme and proposed scheme

It was observed that for any network containing N nodes if the value of t is chosen as t=N/2 +1 then as long as t+1 nodes are compromised the remaining uncompromised nodes will communicate in a secure manner i.e., the network is resilient at t=N/2+1.

Encryption Scheme

For simplicity only modulo 7 and modulo 31 operations are considered during encryption and decryption. So, the key can have combinations of numbers 3 and 5. Initially, the combinations of these matrices ie, 8×8, 16×16, 32×32 are stored in an array and corresponding matrix is chosen depending on input length and key value.

Example 1

Hadamard matrix corresponding to modulo 7

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 6 & 1 & 6 & 1 & 6 & 1 & 6 \\ 1 & 1 & 6 & 6 & 1 & 1 & 6 & 6 \\ 1 & 6 & 6 & 1 & 1 & 6 & 6 & 1 \\ 1 & 1 & 1 & 1 & 6 & 6 & 6 & 6 \\ 1 & 6 & 1 & 6 & 6 & 1 & 6 & 1 \\ 1 & 1 & 6 & 6 & 6 & 6 & 1 & 1 \\ 1 & 6 & 6 & 1 & 6 & 1 & 1 & 6 \end{pmatrix}$$

Hadamard matrix corresponding to modulo 31

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 30 & 1 & 30 & 1 & 30 & 1 & 30 \\ 1 & 1 & 30 & 30 & 1 & 1 & 30 & 30 \\ 1 & 30 & 30 & 1 & 1 & 30 & 30 & 1 \\ 1 & 1 & 1 & 1 & 30 & 30 & 30 & 30 \\ 1 & 30 & 1 & 30 & 30 & 1 & 30 & 1 \\ 1 & 1 & 30 & 30 & 30 & 30 & 1 & 1 \\ 1 & 30 & 30 & 1 & 30 & 1 & 1 & 30 \end{pmatrix}$$

Encryption Part

Input Binary Sequence: 110010011101111110000011

Key {3, 5}

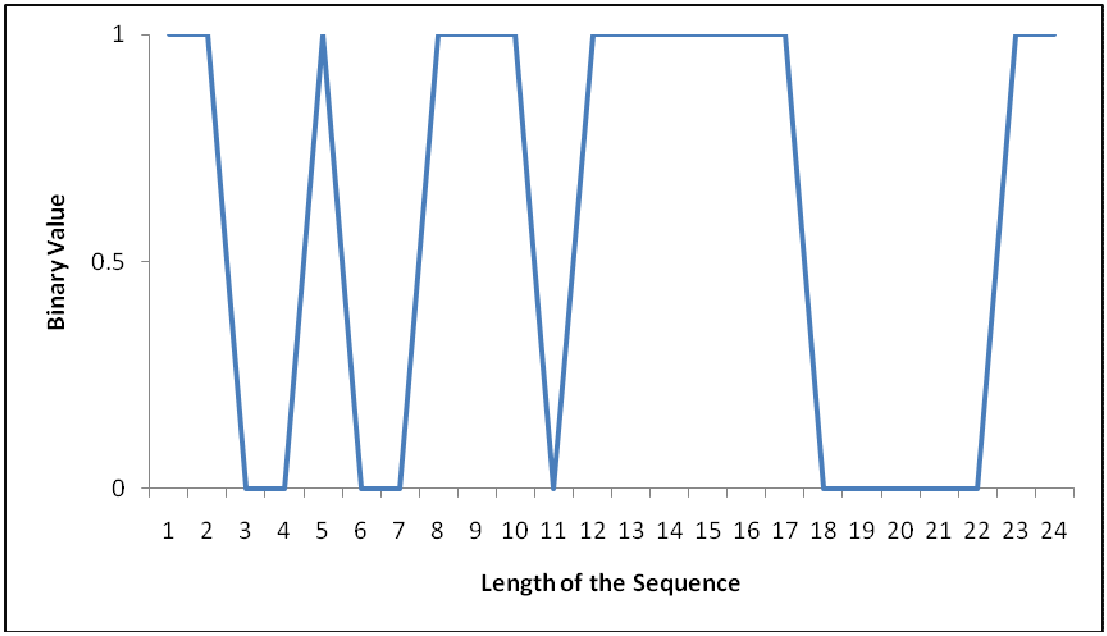

Figure 12. Original Binary Sequence (Example 1)

Level1

At Level 1 every 3 bits in the input sequence is grouped and corresponding decimal value is calculated.

Decimated Sequence: 6, 2, 3, 5, 7, 6, 0, 3

The maximum value corresponding to the input sequence is stored in the array.

A[0][0]={0, 0, 0, 0, 1, 0, 0, 0}

Multiplying Hadamard matrix and decimated sequence and then performing the modulo operation, we get the following sequence

4, 0, 3, 3, 0, 4, 4, 4, 2

Converting to Binary: 100000011011000100100010

Figure 11, Figure 12 and Figure 13 shows length of input sequence, length of sequence at each level and length of encrypted sequence. It can be observed that the length of the sequence has great impact on the number present in the key i.e., for larger values the length of the sequence is very high which makes difficult for adversaries to break the encrypted sequence.

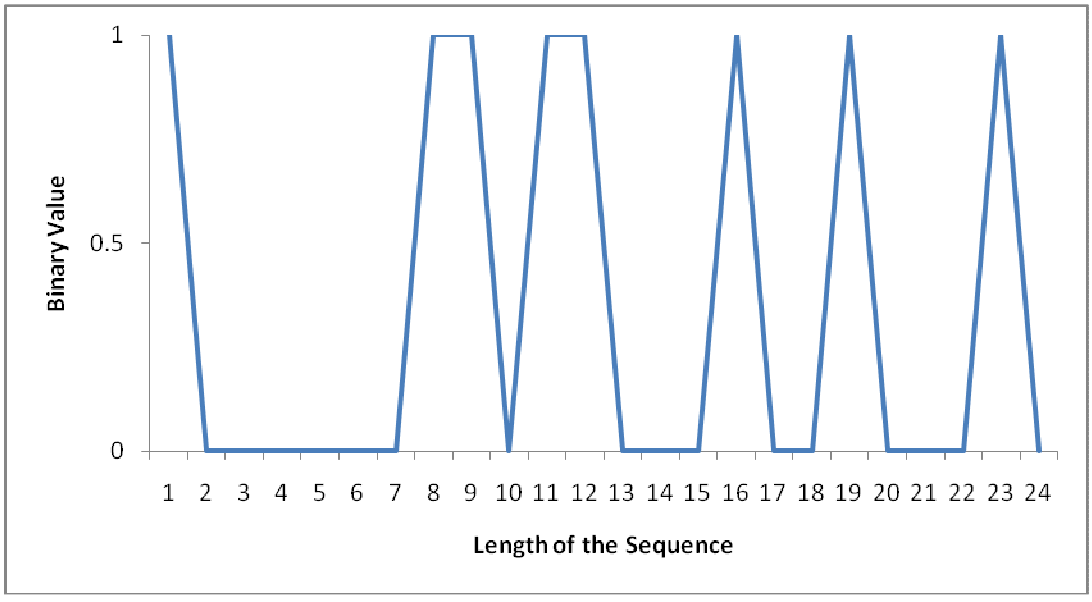

Figure 13. Binary Sequence at Level 1 (Example 1)

Level 2

Modified Binary Sequence: 10000001101100010010001000000000000000000

At Level 2 every 5 bits in the sequence obtained from above level is grouped and corresponding decimal value is calculated.

Decimated Sequence: 16, 6, 24, 18, 4, 0, 0, 0

Multiplying Hadamard matrix and decimated sequence and then performing the modulo operation, we get the following sequence

6, 20, 15, 8, 29, 12, 7, 0

Converting to Binary: 00110101000111101000111010110000011100000

Encrypted Message: 00110101000111101000111010110000011100000

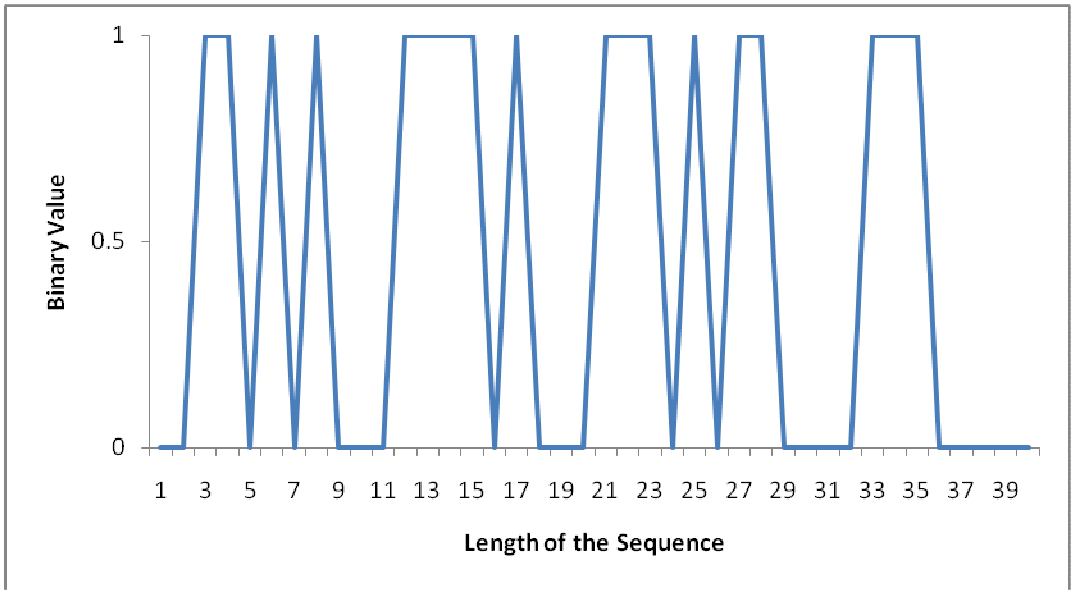

Figure 14. Encrypted Binary Sequence (Example 1)

Figure 8 shows the difference between original sequence and encrypted sequence. By looking at the graph it can be noticed that the encrypted message obtained thereafter is completely different from the original input sequence.

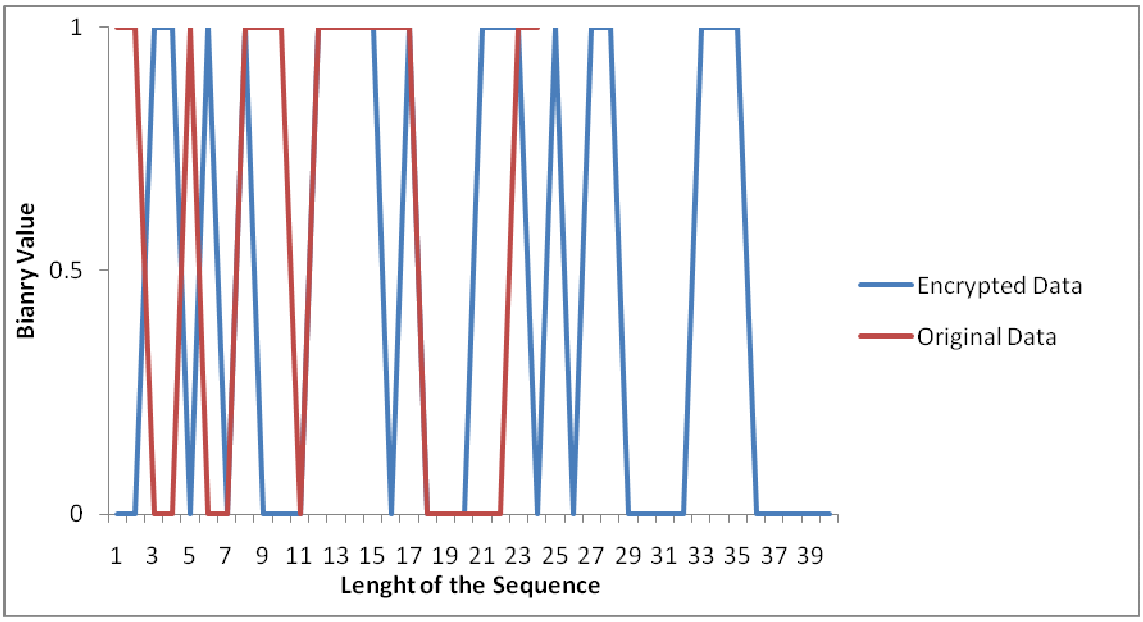

Figure 15. Original Sequence Vs Encrypted Sequence (Example 1)

Decryption Part

Key= {5, 3}

Level1

At Level 1 every 5 bits in the encrypted sequence is grouped and corresponding decimal value is calculated.

Input Sequence: 00110101000111101000111010110000011100000

Decimated Sequence: 6, 20, 15, 8, 29, 12, 7, 0

Multiplying Hadamard matrix and decimated sequence, we get

97, 1257, 967, 1663, 1489, 1953, 1953, 1953

Calculating Modulo Multiplicative Inverse

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 30 & 1 & 30 & 1 & 30 & 1 & 30 \\ 1 & 1 & 30 & 30 & 1 & 1 & 30 & 30 \\ 1 & 30 & 30 & 1 & 1 & 30 & 30 & 1 \\ 1 & 1 & 1 & 1 & 30 & 30 & 30 & 30 \\ 1 & 30 & 1 & 30 & 30 & 1 & 30 & 1 \\ 1 & 1 & 30 & 30 & 30 & 30 & 1 & 1 \\ 1 & 30 & 30 & 1 & 30 & 1 & 1 & 30 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 30 & 1 & 30 & 1 & 30 & 1 & 30 \\ 1 & 1 & 30 & 30 & 1 & 1 & 30 & 30 \\ 1 & 30 & 30 & 1 & 1 & 30 & 30 & 1 \\ 1 & 1 & 1 & 1 & 30 & 30 & 30 & 30 \\ 1 & 30 & 1 & 30 & 30 & 1 & 30 & 1 \\ 1 & 1 & 30 & 30 & 30 & 30 & 1 & 1 \\ 1 & 30 & 30 & 1 & 30 & 1 & 1 & 30 \end{pmatrix} =$$

$$\begin{pmatrix} 8 & 124 & 124 & 124 & 124 & 124 & 124 & 124 \\ 124 & 3604 & 1922 & 1922 & 1922 & 1922 & 1922 & 1922 \\ 124 & 1922 & 3604 & 1922 & 1922 & 1922 & 1922 & 1922 \\ 124 & 1922 & 1922 & 3604 & 1922 & 1922 & 1922 & 1922 \\ 124 & 1922 & 1922 & 1922 & 3604 & 1922 & 1922 & 1922 \\ 124 & 1922 & 1922 & 1922 & 1922 & 3604 & 1922 & 1922 \\ 124 & 1922 & 1922 & 1922 & 1922 & 1922 & 3604 & 1922 \\ 124 & 1922 & 1922 & 1922 & 1922 & 1922 & 1922 & 3604 \end{pmatrix}$$

$$1/8 \times \begin{pmatrix} 8 & 124 & 124 & 124 & 124 & 124 & 124 & 124 \\ 124 & 3604 & 1922 & 1922 & 1922 & 1922 & 1922 & 1922 \\ 124 & 1922 & 3604 & 1922 & 1922 & 1922 & 1922 & 1922 \\ 124 & 1922 & 1922 & 3604 & 1922 & 1922 & 1922 & 1922 \\ 124 & 1922 & 1922 & 1922 & 3604 & 1922 & 1922 & 1922 \\ 124 & 1922 & 1922 & 1922 & 1922 & 3604 & 1922 & 1922 \\ 124 & 1922 & 1922 & 1922 & 1922 & 1922 & 3604 & 1922 \\ 124 & 1922 & 1922 & 1922 & 1922 & 1922 & 1922 & 3604 \end{pmatrix} \text{ mod } 31 =$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

=> a*8 mod 31=1

=> Modulo multiplicative inverse is 4

Matrix after multiplying with modulo multiplicative inverse is

388, 5028, 3868, 6652, 5956, 7812, 7812, 7812

Sequence after performing modulo 31

16, 6, 24, 18, 4, 0, 0, 0

All the 0 values are checked against the array and as there is no maximum value stored, so the sequence is not modified.

16, 6, 24, 18, 4, 0, 0, 0

Converting to Binary: 10000001101100010010001000000000000000000

Level2

At Level 2 every 5 bits in the sequence obtained from previous level is grouped and the corresponding decimal value is calculated.

Discarding Additional 0s: 10000001101100010010010

Decimated Sequence: 4, 0, 3, 3, 0, 4, 4, 2

Multiplying Hadamard matrix and decimated sequence, we get

20, 65, 80, 75, 70, 55, 70, 45

Calculating Modulo Multiplicative Inverse

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 6 & 1 & 6 & 1 & 6 & 1 & 6 \\ 1 & 1 & 6 & 6 & 1 & 1 & 6 & 6 \\ 1 & 6 & 6 & 1 & 1 & 6 & 6 & 1 \\ 1 & 1 & 1 & 1 & 6 & 6 & 6 & 6 \\ 1 & 6 & 1 & 6 & 6 & 1 & 6 & 1 \\ 1 & 1 & 6 & 6 & 6 & 6 & 1 & 1 \\ 1 & 6 & 6 & 1 & 6 & 1 & 1 & 6 \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 6 & 1 & 6 & 1 & 6 & 1 & 6 \\ 1 & 1 & 6 & 6 & 1 & 1 & 6 & 6 \\ 1 & 6 & 6 & 1 & 1 & 6 & 6 & 1 \\ 1 & 1 & 1 & 1 & 6 & 6 & 6 & 6 \\ 1 & 6 & 1 & 6 & 6 & 1 & 6 & 1 \\ 1 & 1 & 6 & 6 & 6 & 6 & 1 & 1 \\ 1 & 6 & 6 & 1 & 6 & 1 & 1 & 6 \end{pmatrix} =$$

$$\begin{pmatrix} 8 & 28 & 28 & 28 & 28 & 28 & 28 & 28 \\ 28 & 148 & 98 & 98 & 98 & 98 & 98 & 98 \\ 28 & 98 & 148 & 98 & 98 & 98 & 98 & 98 \\ 28 & 98 & 98 & 148 & 98 & 98 & 98 & 98 \\ 28 & 98 & 98 & 98 & 148 & 98 & 98 & 98 \\ 28 & 98 & 98 & 98 & 98 & 148 & 98 & 98 \\ 28 & 98 & 98 & 98 & 98 & 98 & 148 & 98 \\ 28 & 98 & 98 & 98 & 98 & 98 & 98 & 148 \end{pmatrix}$$

$$1 \times \begin{pmatrix} 8 & 28 & 28 & 28 & 28 & 28 & 28 & 28 \\ 28 & 148 & 98 & 98 & 98 & 98 & 98 & 98 \\ 28 & 98 & 148 & 98 & 98 & 98 & 98 & 98 \\ 28 & 98 & 98 & 148 & 98 & 98 & 98 & 98 \\ 28 & 98 & 98 & 98 & 148 & 98 & 98 & 98 \\ 28 & 98 & 98 & 98 & 98 & 148 & 98 & 98 \\ 28 & 98 & 98 & 98 & 98 & 98 & 148 & 98 \\ 28 & 98 & 98 & 98 & 98 & 98 & 98 & 148 \end{pmatrix} \text{ mod } 7$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

=> a*1 mod 7=1

=> Modulo multiplicative inverse is 1

Matrix after multiplying with modulo multiplicative inverse

20, 65, 80, 75, 70, 55, 70, 45

Sequence after performing modulo 7

6, 2, 3, 5, 0, 6, 0, 3

All the 0 values are checked against the array and 0 at fifth position is replaced with 7.

Modified Sequence after replacing with maximum value is

6, 2, 3, 5, 7, 6, 0, 3

Converting to Binary: 110010011101111110000011

Decrypted Message: 110010011101111110000011

Example 2

Changing one bit in the decrypted message

Input Sequence: 110010011101111110000011

Encrypted Sequence: 00110101000111101000111010110000011100000

Changed Encrypted Sequence: 10110101000111101000111010110000011100000

Decrypted Sequence: 01001001000111100010000001000001110100000000110

In the example discussed above, the starting bit in the encrypted sequence is changed from 0 to 1and a drastic difference is observed between the original sequence and the decrypted sequence. Further, in addition to the changing of bits in the decrypted part, there is varied length between the sequences.

If the input sequence contains all 1s or 0s then the corresponding encrypted sequence is observed to contain all 0s which implies the input does not carry any useful information.

In the proposed scheme structure of the key plays a vital role. It will be challenging for the eavesdropper to break the sequence if the length of the key is large enough and contains distinct elements. On the other hand if the key length is short or has many duplicate elements then it can be easily broken using brute force or other techniques.

CHAPTER V

CONCLUSION

We have examined that the key generation in original Blom's scheme involves a lot of computation and memory overhead. The previous results helped us to find an alternative way of generating the keys using Blom's scheme. The simulation results of proposed scheme explains that by replacing Vandermonde matrix with Hadamard matrix initial computation overhead can be reduced to a large extent and in later stage of key computation the overhead of storing column of public matrix can be avoided. The original Blom's scheme provides a rough estimation of the secure parameter t and no proper range of t are given, where as the proposed scheme provides a minimum value for the secure parameter t such that the network is more resilient. It is also observed that other decomposition schemes such as LU, LQ and QR may be used to generate a key between any pair of nodes.

As key management is not sufficient to communicate between any pair of nodes, a new approach to encryption is proposed using a chain of Hadamard transforms. In this thesis we implemented the encryption scheme on binary and non-binary message sequence. An algorithm is given for both encryption and decryption which enhances the understanding of the scheme. The simulation results show that this method is more effective when the key size is large and most of the elements in the key are distinct. But, since the chaining of the Hadamard transforms can be as long as one pleases and the formation of blocks that are converted into non-binary numbers is arbitrarily large, the effectiveness of this scheme is potentially very high.

The technique of chaining can also be applied using other transforms. Apart from using the encryption scheme in key management it may be used for hashing by padding the given data block with zeros, finding the encryption sequence and retaining a certain pre-specified number of bits. Such a procedure can then serve as a method of joint encryption and error-correction coding [28].

Further work can be done to investigate the key generation using any of the decomposition schemes. Also, it can be analyzed whether the new scheme is more efficient than Blom's scheme in terms of node connectivity, memory utilization and computation overhead.

REFERENCES

[1] G. J. Pottie and W. J. Kaiser. Wireless Integrated Network Sensors. Communications of the ACM, 43(5):51-58, 2000.

[2] A. T. Butson, Generalised Hadamard matrices, Proc. Amer. Math. Soc.13 (1962) 894–898.

[3] R. Blom, "An Optimal Class of Symmetric Key Generation Systems", In *Advances in Cryptology: EUROCRYPT'84*, LNCS, vol. 209, pp. 335-338, 1985.

[4] L. J. Yan and J. S. Pan, Generalized discrete fractional Hadamard fransformation and its application on the image encryption, International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IEEE Computer Press, pp. 457-460, 2007.

[5] C. C. Gumas, "A century old fast Hadamard transform proves useful in digital communications", Personal Engineering & Instrumentation News, pp. 57-63, November. 1997.

[6] W. Ouyang, W.K. Cham, Fast algorithm for Walsh-Hadamard transform on sliding windows, IEEE Trans. on Pattern Analysis and Machine Intelligence 32, 165-171, 2010.

[7] S. Kak, Classification of random binary sequences using Walsh-Fourier analysis. IEEE Trans. On Electromagnetic Compatibility EMC-13, pp. 74-77, 1971.

[8] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," *Proc. ACM Conf. on Computer and Commun. Security*, pp. 41–47, November. 2002.

[9] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," *Proc. IEEE Symp. on Security and Privacy*, pp. 197-213, May 2003.

[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," *Proc. IEEE Conf. on Computer Commun.* (*INFOCOM*), pp. 586-597, March. 2004.

[11] Y. Zhou and Y. Fang, "A two-layer key establishment scheme for wireless sensor networks," *IEEE Trans. Mobile Computing*, vol. 6, no. 9, pp. 1009-1020, September. 2007.

[12] D. Liu and P. Ning, "Location-based pairwise key establishments for relatively static sensor networks," *Proc. ACM Workshop on Security of Ad Hoc and Sensor Networks* (*SASN*), pp. 72-82, October. 2003.

[13] Z. Yu and Y. Guan, "A key management scheme using deployment knowledge for wireless sensor networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, no. 10, pp. 1411-1425, October. 2008.

[14] Y. Zhou, Y. Zhang, and Y. Fang, "Key establishment in sensor networks based on triangle grid deployment model," *Proc. IEEE Military Commun. Conf.* (*MILCOM*), pp. 1450- 1455, October. 2005.

[15] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung. ”Perfect-secure key distribution of dynamic conferences”, In *Advances in Cryptography*– CRYPTO ’92, LNCS 740, pp. 471-486, 1993.

[16] N. T. Canh, Y. K. Lee, and S. Y. Lee, “HGKM - A Group-based key management scheme for sensor networks using deployment knowledge,” *Sixth Annual Conference on Communication Networks and Services Research (CNSR ’08)*, 5-8 May 2008.

[17] W. H. Press, B.P. Flannery, S. A. Teukolsky and W. T. Vetterling, “Vandermonde Matrices and Toeplitz Matrices." 2.8 in *Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed*. Cambridge, England: Cambridge University Press, pp. 82-89, 1992.

[18] M. Gnanaguruparan and S. Kak, Recursive hiding of secrets in visual cryptography. Crypto logia 26: 68-76, 2002.

[19] A. Parakh and S. Kak, Online data storage using implicit security. Information Sciences 179:3323-3331, 2009.

[20] A. Parakh and S. Kak, Space efficient secret sharing for implicit data security. Information Sciences 181:335-341, 2011.

[21] A. Parakh and S. Kak, Efficient key management in sensor networks. Proceedings IEEE GLOBECOM, pp. 1584-1589, Miami, 2010.

[22] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti. Secure publications. *Proceedings of the 2001 ACM International Symposium on Mobile AdHoc Networking and Computing, MobiHoc 200*1, pp 156-163, October 2001.

[23] W. Du, J. Deng, Y Han, and P. Varshney. A pairwise key pre distribution scheme for wireless sensor networks. In *In Proceedings of the Tenth ACM Conference on Computer and Communications Security (CCS 2003)*, pp 42-51, October. 2003.

[24] W. Du, J. Deng, Y S. Han, P. Varshney, J. Katz, and A. Khalili. A pairwise key pre distribution scheme for wireless sensor networks. In *ACM Transactions on Information and System Security (TISSEC)*, 8(1), 41–77, 2005.

[25] D. Liu, P. Ning, and R. Li. Establishing pairwise keys in distributed sensor networks. In *ACM Trans. Inf Syst. Secur*., 8:41-77, October 2005.

[26] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar, “Spins: Security protocols for sensor networks,” *Wireless Networks Journal (WINE)*, No.8, pp. 521-534, September 2002.

[27] S. Zhu, S. Setia, and S. Jajodia, “LEAP: efficient security mechanisms for large-scale distributed sensor networks,” In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 62–72, Washington, DC, USA, October 2003.

[28] S. Kak, Joint encryption and error-correction coding, Proceedings of the 1983 IEEE Symposium on Security and Privacy, pp. 55-60 1983.

[29] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next Century Challenges: Mobile Networking for "Smart Dust". In *Proceeding of the 5th ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp 271-278, Seattle, Washington, USA, August 1999.

[30] L. F. Akyildiz,W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communication Magazine*, 40(8):102-114, August 2002.

[31] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, 8(5):521-534, September 2002.

[32] A. D. Wood and J. A. Stankovic. Denial of Service in Sensor Networks. *IEEE Computer Magazine*, 35(10):54-62, October 2002.

[33] H.T. Kung and D. Vlah. Efficient Location Tracking Using Sensor Networks. In *Proceeding of IEEE Wireless Communications and Networking Conference (WCNC)*, pp 1954-1961, New Orleans, LA, USA, March 2003.

[34] R. R. Brooks, P. Ramanathan, and A. M. Sayeed. Distributed Target Classification and Tracking in Sensor Networks. *Proceedings of the IEEE*, 91(8):1163-1171, August 2003.

VITA

Singi Reddy Rohith Reddy

Candidate for the Degree of

Master of Science

Thesis: KEY MANGEMENT AND ENCRYPTION IN WIRELESS SENSOR NETWORKS USING HADAMARD TRANSFORMS

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2011.

Completed the requirements for the Bachelor of Engineering in Computer Science and Engineering at Jawaharlal Nehru Technological University, Hyderabad, India in 2009.

Experience:

Research Assistant *Oklahoma State University, Stillwater, OK,* December 2009 to present

Projects:

- Grand River Dam Authority (Dec 2009 – July 2010)
- Oklahoma Landmarks Inventory Database (Aug 2010 – Feb 2011)
- National Register of Historic Places in Oklahoma (Feb 2011 – May 2011)

Developed and maintained database application for each of the project using asp.net, SQL server 2005 and SQL server 2008.

Graduate Assistant *Oklahoma State University, Stillwater, OK,* September 2009 to December 2009

- Developed and maintained Entrepreneurship Department website using joomla 1.0.
- Helped the department in organizing several events.

Name: Singi Reddy Rohith Reddy Date of Degree: July, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: KEY MANAGEMENT AND ENCRYPTION IN WIRELESS SENSOR NETWORKS USING HADAMARD TRANSFORMS

Pages in Study: 41 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study:

This thesis deals with key management in wireless sensor networks. The shortcomings of existing schemes include poor resilience against node capture, low secure connectivity and memory overhead. After analyzing the problems in existing key management schemes, a modified model of the Blom's scheme is proposed in this thesis. The new model helps to overcome the prominent problems such as memory cost and computation overhead. The modified key management scheme not only enhances the efficiency of the existing Blom's scheme but also provides a new way of generating keys in wireless sensor networks. In our work, we also propose a new encryption technique which can be used in wireless sensor networks to provide security to the data. The encryption technique can be used either on binary or non-binary data and the chaining strategy used in this scheme makes it difficult for intruders to break the message.

Findings and Conclusions:

Based on the analysis and simulations, a new value is given to the secure parameter t of Blom's scheme. It is observed that the Hadamard matrix used in the modified scheme helps to reduce the computation, communication and memory overhead in the generation of keys. The simulation results of the proposed encryption scheme shows that length of the encrypted message can be varied and the length in turn is directly proportional to the values in the key. In addition, the security of the scheme depends on the depth of the key i.e., number of elements present in the key.

ADVISER'S APPROVAL: Dr. Subhash Kak