International Journal of Engineering Applied Sciences and Technology, 2022
Vol. 7, Issue 11, ISSN No. 2455-2143, Pages 138-142
Published Online March 2023 in IJEAST (http://www.ijeast.com)

LIGHT WEIGHT SECURITY ALGORITHM FOR LOW POWER IOT DEVICE

DR.K.N.S Lakshmi, Y.Jyothika sai ramya
Department of computer science
SVPEC, visakhapatnam, Andhra pradesh,
India

Abstract—**Many IOT devices(mainly networks of IOT devices or networks that are deployed to monitor calamity situations) are deployed in an arbitrary and unplanned fashion. For any sensor in such a network can end up being adjacent node to any other IOT node in the network. To Establish a secure communication between every pair of adjacent IOTs node in such a network, each sensor node x in the network needs to store n -1 number of symmetric keys that sensor node x shares with all the other sensor nodes, where n is the number of sensor nodes in the present network. This memory storage requirement of the keying protocol is various, especially when n is large and the available storage in each sensor node is modest. Previous efforts to redesign this keying protocol and reduce the number of keys to be stored in each sensor node have produced protocols that are vulnerable to impersonation, eavesdropping, and collusion attacks. In this paper, we present a fully secure protocol.**

Keywords— **IOT, key sender, encryption, decryption, IOT networks, MWSN, plain-text, cipher-text, grid, probabilistic, keying protocol**

I. INTRODUCTION

When a pair of devices wants to communicate with each other, there has to be a secure connection that needs to be established between them. The same is the case with the IOT nodes. When two IOT nodes tend to communicate with each other the data that is being communicated has to be protected from external attacks.

IOT nodes are very small devices with small size, small computation power and transmission range. Moreover, the positions of IOTs need not be static; they may be dynamically displaced with respect to time. So, it is imperative that the data that needs to be communicated is kept accurate and secure. In early days when IOTs were first introduced there were many problems related to security. Either the data that was being communicated was too large or the secure transmission aspect was neglected.

II. PROPOSED ALGORITHM

A. **Watermark embedding algorithm –**

In case of two-dimensional image, after a DWT transform, the image is divided into four corners, upper left corner of the original image, lower left corner of the vertical details, upper right corner of the horizontal details, lower right corner of the component of the original image detail (high frequency). You can then continue to the low frequency components of the same upper left corner of the 2nd, 3rd inferior wavelet transform.

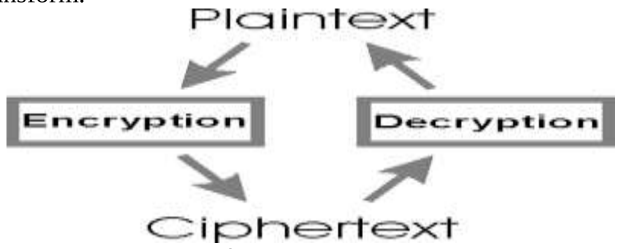


Figure 1: Conversion of plain text to cipher text and vice versa

There are two main protocols that were proposed in the past to reduce the number of stored keys in each IOT in the network. We refer to these two protocols as **the Probabilistic Keying Protocol and the Grid Keying Protocol**. number of keys that are selected at random from a large set of keys. When two adjacent IOTs need to exchange data messages, the two IOTs identify which keys they have in common then use a combination of their common keys as a symmetric key to encrypt and decrypt their exchanged data messages. Since a particular IOT doesn't have its own universal key and only has a set of shared keys it is prone to impersonation attack**.** In the **Grid Keying Protocol**, each IOT is allocated a unique number (called identifier) which is the coordinates of a distinct node in a two-dimensional space and each symmetric key is also assigned an identifier. Then a IOT x stores a symmetric key K if and only if the identifiers of x and K satisfy certain given relation. When two adjacent IOTs need to communicate, the two IOTs identify which keys they have in common then use a combination of those common keys as a symmetric key to encrypt and decrypt data messages. The grid keying protocol has two advantages). First, this protocol can defend against impersonation (unlike the probabilistic protocol) and can defend against eavesdropping (like the probabilistic protocol). Second, each IOT in this protocol needs to store only O(log n) symmetric keys, where n is the number of IOTs in the network. But the main problem with grid keying protocol is it cannot defend itself from collusion attack.

In our proposed paper we show that there can be a system with a keying protocol which reduces the number of keys maintained within the IOT to (n+1)/2 keys. The additional and a very important feature that has been introduced is that of a **Key Sender.** Each and every IOT Network has a Key Sender associated with it. Every IOT node needs to get registered within the Key Sender. The Key Sender then distributes the keys to the IOTs within the network. With the help of the keys distributed by the Key Sender the IOTs communicate with the other IOTs.

We use ix and I_y to denote the identifiers of IOTs 'x' and 'y', respectively, in this network. Each two IOTs, say IOT nodes 'x' and 'y', share a symmetric key denoted $K_{(x, y)}$ or $K_{(y, x)}$. Only the two IOT nodes 'x' and 'y' know their shared key $K_{(y, x)}$ [7]. And if IOT nodes 'x' and 'y' ever become neighbors in the network, then they can use their shared symmetric key $K_{(y, x)}$ to perform two functions:

1) **Mutual Authentication**: IOT node 'x' authenticates IOT node 'y', and IOT node y authenticates IOT node 'x'.

2) **Confidential Data Exchange**: Encrypt and later decrypt all the exchanged data messages between 'x' and 'y'. In the remainder of this section, we show that if the shared symmetric keys are designed to have a "special structure", then each IOT node needs to store only (n+1)/2 shared symmetric keys. But before we present the special structure of the shared keys, we need to introduce two new concepts: "**Universal Keys**" and "a circular relation, named below, over the IOT node identifiers". Each IOT node 'x' in the network stores a symmetric key, called the universal key of IOT node 'x'. The universal key of IOT node 'x', denoted 'u_x', is known only to IOT node 'x'. Let I_x and iy be two distinct IOT node identifiers. Identifier ix is said to be below identifier I_y if exactly one of the following two conditions holds:

1) $I_x<I_y$ and $(I_y - I_x) < n/2$

2) $I_x>I_y$ and $(I_x - I_y) > n/2$

The below relation is better explained by an example. Consider the case where n / 3. In this case, the IOT node identifiers are 0, 1, 2

We have:

- Identifier 0 is below identifiers 1 and 2.
- Identifier 1 is below identifiers 2 and 0.
- Identifier 2 is below identifiers 1 and 0.
-

1. Methodology

Theorem 1: If there exists a pair of distinct but adjacent IOT nodes 'x' and 'y' with unique identifiers 'I_x' and 'I_y' respectively then the Below condition holds true as follows :-

- 'I_x' is below 'I_y'
- 'I_y' is below 'I_x'

Theorem 2: Since there exists 'n IOT nodes, each IOT node 'x' with identifier ix has (n-1)/2 IOT node identifiers I_y below it.

Theorem 3: In accordance with Theorem 1,the number of IOT nodes with identifiers ix below the IOT node 'y' with identifier I_y is (n-1)/2.

Theorem 4: If a IOT node identifier ix for IOT node 'I_x' is below a IOT node identifier 'I_y' then the IOT node 'x' needs to store the Symmetric Key '$k_{y, x}$'/ H ('I_x'|u_y) within it. Then the IOT node 'y' needs to compute the Symmetric Key to verify the IOT node 'x'. The Symmetric Key '$k_{(y, x)}$' is stored only in 'x'.

Theorem 5: As discussed earlier, each IOT node 'x' needs to store single Universal Key and (n-1)/2 Symmetric Keys '$k_{(y, x)}$' in order to communicate with IOT node 'y' (NOTE: the IOT node identifier ix should be below 'I_y').

3. A Mutual Authentication Protocol:

Each and every IOT node 'x' is provided with the following information before the IOT nodes are deployed within the network:-

1) One distinct identifier ix in the range 0-(n-1)

2) One universal key u_x

3) (n-1)/2 symmetric keys $K_{(y, x)}$ / H ('I_x'|u_y) each of which is shared between IOT node 'x' and another IOT node 'y' (where ix is below I_y). If the IOT nodes 'x' and 'y' are adjacent and want to communicate with each other, then they must implement the Mutual Authentication protocol which has the following steps :-

Step 1: IOT node 'x' selects a random nonce n_x and sends a hello message that is received by IOT node 'y'. x ->y: hello('I_x'|n_x)

Step 2: IOT node 'y' selects a random nonce n_y and sends a hello message that is received by IOT node 'x'. x ->y: hello ('I_y'|n_y)

Step 3: IOT node 'x' determines whether ix is below iy. Then it either fetches $K_{(y, x)}$ from its memory or computes it. Finally, IOT node 'x' sends a verify message to IOT node 'y'. x->y: verify ('I_x'; 'I_y'; H (I_x | I_y |n_y|$K_{(y, x)}$))

Step 4: IOT node 'y' determines whether iy is below ix. Then it either fetches $K_{y,x}$ from its memory or computes it. Finally, IOT node 'y' sends a verify message to IOT node 'x'. x->y: verify (I_y| I_x |H ((I_y| I_x |n_x|$K_{y,x}$))

Step 5: IOT node 'x' computes H (I_y| I_x |n_x|$K_{(y, x)}$) and compares it with the received H(I_y|I_x|n_x|$K_{(y, x)}$). If they are equal, then IOT node 'x' concludes that the IOT node claiming to be IOT node 'y' is indeed IOT node 'y'. Otherwise, no conclusion can be reached.

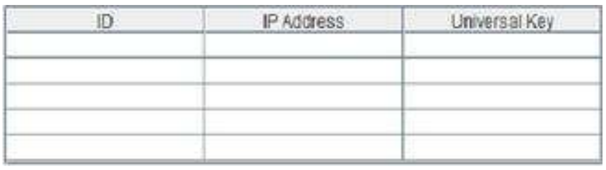
Step 6: IOT node 'y' computes H(I_x | I_y |n_y|$K_{(y, x)}$) and compares it with the received H(I_x | I_y |n_y|$K_{(y, x)}$). If they are equal, then 'y' concludes that the IOT node claiming to be IOT node 'x' is indeed IOT node 'x'. Otherwise, no conclusion can be reached.

4. A Data Exchange Protocol:
IOT nodes ‘x’ and ‘y’ can now start exchanging data according to the following data exchange protocol:-
Step 1: IOT node ‘x’ combines the nonce n_y with the data to be sent, encrypts the combined data using the symmetric key $K_{(y, x)}$, and sends the result in a data message to IOT node ‘y’.
x ->y: data(I_x | I_y |$K_{(y, x)}$(n_y|text))
Step 2: IOT node ‘y’ combines the nonce n_x with the data to be sent, encrypts the combined data using the symmetric key $K_{y,x}$, and sends the result in a data message to IOT node ‘x’.
x ->y: data(I_y | I_x |$K_{(y, x)}$(n_x|text))

5. Optimality of Keying Protocol:
According to our keying protocol, described in Section III, each IOT node in the network is required to store only (n+1)/2 keys. Thus, the total number of keys that need to be stored within the network is n(n+1)/2.
Theorem 6: There should be a minimum of n(n-1)/2 keys that are to be stored within the IOT node network.
Theorem 7: According to any keying protocol (which is uniform) has to store at least (n-1)/2 keys within it to communicate with its adjacent IOT nodes.

6. Triple Data Encryption Algorithm (TDEA):
DES (the Data Encryption Standard) is a symmetric block cipher developed by IBM. The algorithm uses a 56-bit key to encipher/decipher a 64-bit block of data. The key is always presented as a 64-bit block, every 8th bit of which is ignored. However, it is usual to set each 8th bit so that each group of 8 bits has an odd number of bits set to 1.
The algorithm is best suited to implementation in hardware, probably to discourage implementations in software, which tend to be slow by comparison. However, modern computers are so fast that satisfactory software implementations are readily available.
DES is the most widely used symmetric algorithm in the world, despite claims that the key length is too short. Ever since DES was first announced, controversy has raged about whether 56 bits is long enough to guarantee security.
The key length argument goes like this. Assuming that the only feasible attack on DES is to try each key in turn until the right one is found, then 1,000,000 machines each capable of testing 1,000,000 keys per second would find (on average) one key every 12 hours. Most reasonable people might find this rather comforting and a good measure of the strength of the algorithm.
Those who consider the exhaustive key-search attack to be a real possibility (and to be fair the technology to do such a search is becoming a reality) can overcome the problem by using double or triple length keys. In fact, double length keys have been recommended for the financial industry for many years.
Use of multiple length keys leads us to the Triple-DES algorithm, in which DES is applied three times. If we consider a triple length key to consist of three 56-bit keys K1, K2, K3 then encryption is as follows:
• Encrypt with K1
• Decrypt with K2
• Encrypt with K3
Decryption is the reverse process:
• Decrypt with K3
• Encrypt with K2
• Decrypt with K1
Setting K3 equal to K1 in these processes gives us a double length key K1, K2. Setting K1, K2 and K3 all equal to K has the same effect as using a single-length (56-bit key). Thus it is possible for a system using triple-DES to be compatible with a system using
single-DES.

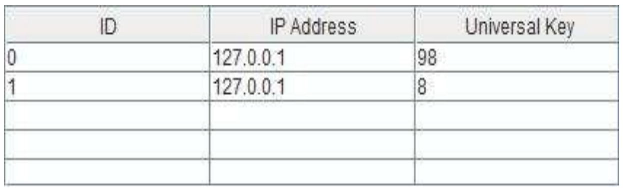
7. Data Analysis
Key Sender: It detects the IOT nodes present in its regionand updates their details in its table. Here we used java simulation to create an interface to key-sender. Symbolically it may look like(before IOT nodes detection and after detection)



ID	IP Address	Universal Key

Figure3: KeySender table before detection

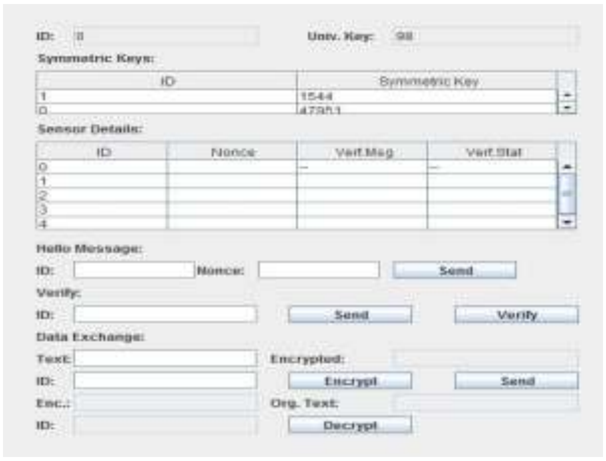
ID is the unique number given to each IOT node, IP address is the logical address and universal is the symmetric key used for encryption and decryption
Key Sender Table after IOT node detection:



ID	IP Address	Universal Key
0	127.0.0.1	98
1	127.0.0.1	8

Figure4: Key-sender table after clients/IOT nodes are detected

Here we take 2 IOT node nodes or clients (say receiver 0 and receiver 1) which are detected by the key sender. The key sender then calculates the universal keys(as shown in fig-4) and sends them to the respective clients



ID: 0 Univ. Key: 98

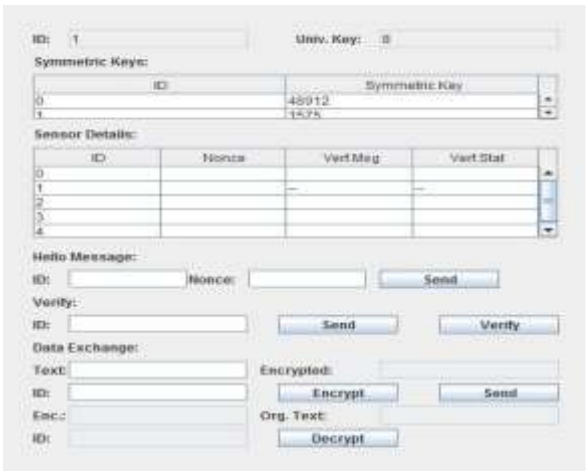
Symmetric Keys:

ID	Symmetric Key
1	1544
0	47951

Sensor Details:

ID	Nonce	Verf.Msg	Verf.Stat
0		--	--
1			
2			
3			
4			

Figure5:Receiver0



ID: 1 Univ. Key: 8

Symmetric Keys:

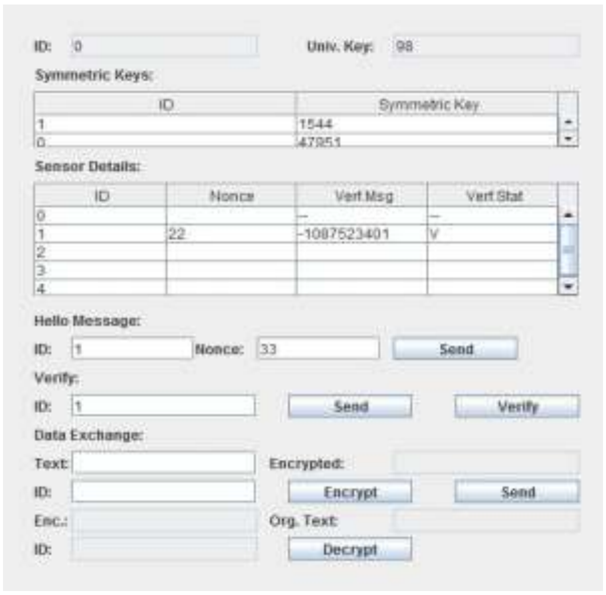
ID	Symmetric Key
0	48912
1	1575

Sensor Details:

ID	Nonce	Verf.Msg	Verf.Stat
0			
1		--	--
2			
3			
4			

Figure6:Receiver1

After the key sender sends the symmetric keys to both the receivers the clients now look like



ID: 0 Univ. Key: 98

Symmetric Keys:

ID	Symmetric Key
1	1544
0	47951

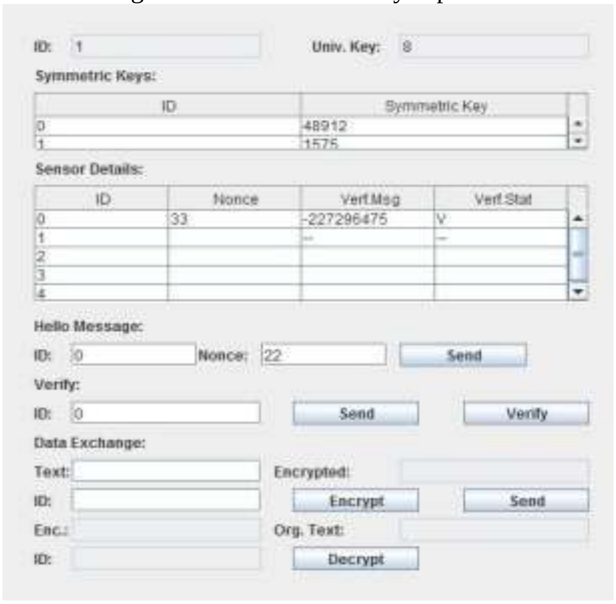
Sensor Details:

ID	Nonce	Verf.Msg	Verf.Stat
0		--	--
1	22	-1087523401	V
2			
3			
4			

Hello Message: ID: 1 Nonce: 33

Verify: ID: 1

Figure7:Reciever0 with keys updated



ID: 1 Univ. Key: 8

Symmetric Keys:

ID	Symmetric Key
0	48912
1	1575

Sensor Details:

ID	Nonce	Verf.Msg	Verf.Stat
0	33	-227296475	V
1		--	--
2			
3			
4			

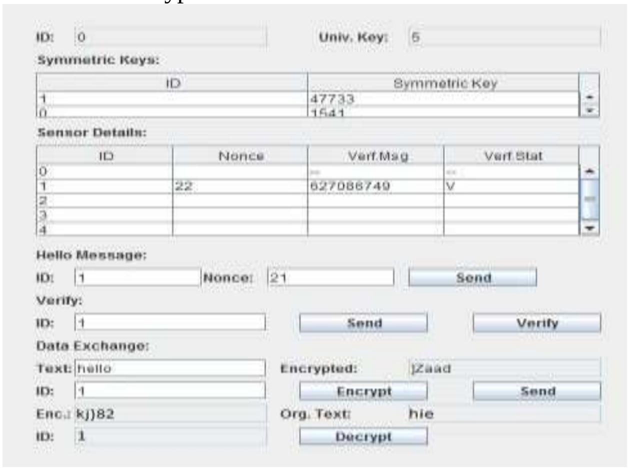
Hello Message: ID: 0 Nonce: 22

Verify: ID: 0

Figure8:Receiver1 with keys updated

After the keys are updated both the nodes need to authenticate each other for that they send verification messages and confirm the authentication. After authentication is done message transfer is done using encryption and decryption.

Message transfer done by Receiver0-original message is hello, it is encrypted and sent. The encrypted message from Receiver1 is decrypted



ID: 0 Univ. Key: 5

Symmetric Keys:

ID	Symmetric Key
1	47733
0	1541

Sensor Details:

ID	Nonce	Verf.Msg	Verf.Stat
0		--	--
1	22	627086749	V
2			
3			
4			

Hello Message: ID: 1 Nonce: 21

Verify: ID: 1

Data Exchange:

Text: hello Encrypted: |Zaad

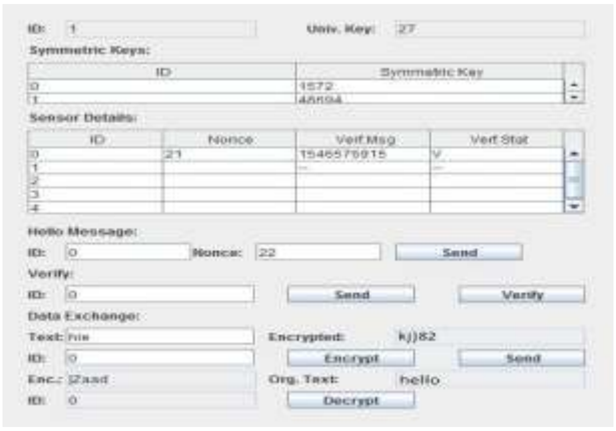
ID: 1

Enc.: kj)82 Org. Text: hie

ID: 1

Figure9: Receiver0 sending and receiving messages

Message transfer done by Receiver1-original message is hie, it is encrypted and sent. The encrypted message from Receiver0 is decrypted


ID: 1 Univ. Key: 27

Symmetric Keys:

ID	Symmetric Key
0	1572
1	45594

Sensor Details:

ID	Nonce	Verf Msg	Verf Stat
0	21	1546576915	V
1		--	--
2			
3			
4			

Hello Message:
ID: 0 Nonce: 22 Send

Verify:
ID: 0 Send Verify

Data Exchange:
Text: hie Encrypted: kj)82
ID: 0 Encrypt Send
Enc.: |2aad Org. Text: hello
ID: 0 Decrypt

Figure10: Receiver1 sending and receiving messages

Whenever a receiver0 wants to communicate with receiver2, it cannot communicate directly, first of all the receiver0 must communicate with receiver1 and then the receiver1 communicates that message with receiver2

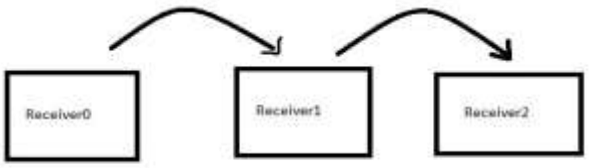


Figure9: nodes communication

III. CONCLUSION

Typically, each sensor in a sensor network with n IOT's needs to store n - 1 shared symmetric keys to communicate securely with each other. Thus, the number of shared symmetric keys stored in the sensor network is n(n - 1). However, the optimal number of shared symmetric keys for secure communication, theoretically, is (n 2) = n(n - 1)/2. In this paper, we show the Secure minimal or Light weight Key Agreement protocol for sensor networks, that needs to store only (n + 1)/2 shared symmetric keys to each sensor. The number of shared symmetric keys stored in a sensor network with n IOT's is n(n + 1)/2, which is close to the optimal number of shared symmetric keys for any key distribution scheme that is not vulnerable to collusion.

Firstly, it is uniform: we store the same number of keys in each sensor. Secondly, it is computationally cheap, and thus suitable for a low-power computer such as a sensor: when two IOT's are adjacent to each other, the computation of a shared symmetric key requires only hashing.

IV. REFERENCES

[1]. [I] Communication within IOT node Networks by Using Key Distributor by ch.d.naiduijcsit 2014.

[2]. The remaining references are:

[3]. Taehwan Choi, H. B. Acharya and Mohamed D. Gouda, " The Best Keying Protocol ", December 2011 IEEE

[4]. http://en.wikipedia.org/wiki/IOT node -Sensors

[5]. Sensor Networks by Margaret Rouse.

[6]. http://en.wikipedia.org/wiki/Mobilewirelesssensornetwork- MWSN's

[7]. "Key Distribution Mechanisms for Wireless Sensor Networks" by Seyit A., C¸Amtepe and BulentYener

[8]. "Cryptography and Network Security", Fourth Edition by William Stallings

[9]. L. Gong and D. J. Wheeler, "A matrix key-distribution scheme," Journal of Cryptology, vol. 2, pp. 51–59, January 1990.

[10]. "Advanced Encryption Standard" by Douglas Selent

[11]. http://en.wikipedia.org/wiki/Advanced_Encryption_Standard#cite_note-fips-197-4.

[12]. "Comparative Study of Energy Aware QoS for Proactive and Reactive Routing Protocols for Mobile Ad-hoc Networks". International Journal of Computer Applications (0975 – 8887) Volume 31– No.5, October 2011.

[13]. Steganography Detection using Functional Link Artificial Neural Networks, International Journal of Computer Applications (0975 - 888), Volume 47 No.5 June 2012.

[14]. Secure Group Communication using Multicast Key Distribution Scheme in Ad-hoc Network, International Journal of Computer Applications. (0975 - 8887)Volume 1 – No. 25, Nov-2010.