ISSN:0975-9646

A Novel Secured Data and Key Exchange Mechanism for Sensor Networks

Bhargavi
M.Tech Student
Dept of CSE, AIET

Vasupalli Mahesh
Asst. Professor,
Dept of CSE, AIET

Y.Rameshkumar
Asst. Professor,
Dept of CSE, AIET

Abstract: A sensor in order to communicate with another sensor within the sensor network needs to find out the sensor which is adjacent to it. So within the network, a sensor needs to find out its adjacent sensor and communicate with it. In order to communicate it needs to maintain some authentication information. The information must be transferred securely. If the sensors are mobile then it is difficult to expect the adjacent sensors. So each sensor has to maintain the information of all other sensors. This increases the non-data overhead. So, if there are 'n' sensors within the network then each sensor has to maintain the keys of the remaining 'n-1' sensors (which is 'n*(n-1)') for the network. So through our project we reduce the keys information from 'n-1' to a maximum of '(n+1)/2'. This could be achieved with the help of a new concept called 'Key Sender. In this paper we are used TDEA algorithm for the encryption purpose.

Keywords: sensor, key sender, encryption, decryption, sensor networks, MWSN, plain-text, cipher-text, grid, probabilistic, keying protocol

1. 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 sensor nodes. When two sensor nodes tend to communicate with each other the data that is being communicated has to be protected from external attacks.

Sensor nodes are very small devices with small size, small computation power and transmission range. Moreover, the positions of sensors 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 sensors 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.

Therefore the concept of Cryptography was introduced. Cryptography ensured that the data that is being communicated is secure and also the data size is also small.

Sensors: Sensors are defined as devices that measure a physical quantity and convert it into an electric signal which can be read and understood by an observer or an instrument [2]. It is generally used to communicate secretive information securely without the attack from external world. Examples: SONAR (Battle fields), Heat Sensors, Security Alarm Systems.

Sensor Networks: A sensor network is a group of specialized sensors with a communication infrastructure intended to monitor and record conditions at diverse locations. A sensor network consists of multiple detection stations called sensor nodes, each of which is small, lightweight and portable. Every sensor node is equipped with a transducer, and power source. The transducer generates electrical signals based on sensed physical effects and phenomena [3]. The microcomputer processes and stores the sensor output. The power for each sensor node is derived from the electric utility or from a battery.

Mobile wireless sensor networks (MWSNs):

MWSNs can simply be defined as a wireless sensor network (WSN) in which the sensor nodes are mobile [10]. MWSNs are a smaller, emerging field of research in contrast to their well-established predecessor. Many of their applications are similar, such as environment monitoring or surveillance [4] .

Key distribution within Sensor Networks:

Itis an important issue in wireless sensor network (WSN) design. Due to memory and power constraints, the keys need to be well arranged to build a fully functional network.

Key distribution [12]is the method of distribution of keys to nodes before deployment. Initially sensors detect their adjacent sensors in the network and send that information to the key sender. The Key Sender then generates secret keys and sends them to respective sensors for secure transmission [5].

Encryption and Decryption of data:

In cryptography, **Encryption** is the process of encoding messages or information in such a way that only authorized parties can read it. In an encryption scheme, the message or information, referred to as plain-text, is encrypted using an encryption algorithm [11], turning it into an unreadable cipher text [6].

Decryption: It is the process of decoding the data which has been encrypted into a secret format. An authorized user can only decrypt data because decryption requires a secret key or password. In simple terms it is the conversion of cipher text into plain text [6].

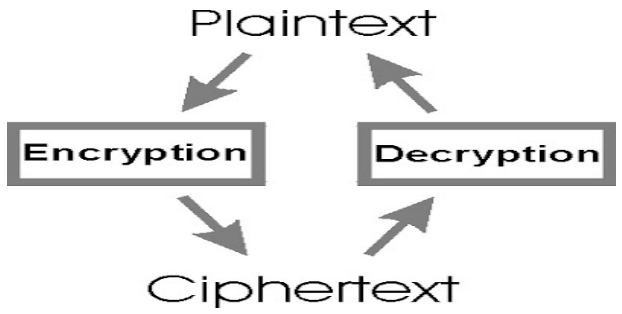


Figure 1: Conversion of plain text to cipher text and vice versa

There are two main protocols that were proposed in the past to reduce the number of stored keys in each sensor in the network. We refer to these two protocols as **the Probabilistic Keying Protocol and the Grid Keying Protocol**. In the **Probabilistic Keying Protocol** [1], each sensor in the network stores a number of keys that are selected at random from a large set of keys. When two adjacent sensors need to exchange data messages, the two sensors 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 sensor 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 sensor 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 sensor x stores a symmetric key K if and only if the identifiers of x and K satisfy certain given relation. When two adjacent sensors need to communicate, the two sensors 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 sensor in this protocol needs to store only O(log n) symmetric keys, where n is the number of sensors 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 sensor 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 Sensor Network has a Key Sender associated with it. Every sensor node needs to get registered within the Key Sender. The Key Sender then distributes the keys to the sensors within the network. With the help of the keys distributed by the Key Sender the sensors communicate with the other sensors.

We use ix and I_y to denote the identifiers of sensors 'x' and 'y', respectively, in this network. Each two sensors, say sensors 'x' and 'y', share a symmetric key denoted $K_{(x, y)}$ or $K_{(y, x)}$. Only the two sensors 'x' and 'y' know their shared key $K_{(y, x)}$ [7]. And if sensors '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**: Sensor 'x' authenticates sensor 'y', and sensor y authenticates sensor '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 sensor 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 sensor identifiers". Each sensor 'x' in the network stores a symmetric key, called the universal key of sensor 'x'. The universal key of sensor 'x', denoted 'u_x', is known only to sensor 'x'. Let I_x and iy be two distinct sensor 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 sensor 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.

2. METHODOLOGY

Theorem 1: If there exists a pair of distinct but adjacent sensors '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 sensors, each sensor 'x' with identifier ix has (n-1)/2 sensor identifiers I_y below it.

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

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

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

3. A MUTUAL AUTHENTICATION PROTOCOL:

Each and every sensor 'x' is provided with the following information before the sensors 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 sensor 'x' and another sensor 'y' (where ix is below I_y). If the sensors '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: Sensor 'x' selects a random nonce n_x and sends a hello message that is received by sensor 'y'. x ->y: hello('I_x'|n_x)

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

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

Step 4: Sensor 'y' determines whether iy is below ix. Then it either fetches $K_{y,x}$ from its memory or computes it.

Finally, sensor ‘y’ sends a verify message to sensor ‘x’. x->y: verify (I_y| I_x |H ((I_y| I_x |n_x|$K_{y,x}$))

Step 5: Sensor ‘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 sensor ‘x’ concludes that the sensor claiming to be sensor ‘y’ is indeed sensor ‘y’. Otherwise, no conclusion can be reached.

Step 6: Sensor ‘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 sensor claiming to be sensor ‘x’ is indeed sensor ‘x’. Otherwise, no conclusion can be reached.

4. A DATA EXCHANGE PROTOCOL:

Sensors ‘x’ and ‘y’ can now start exchanging data according to the following data exchange protocol:-

Step 1: Sensor ‘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 sensor ‘y’.

x ->y: data(I_x | I_y |$K_{(y, x)}$(n_y|text))

Step 2: Sensor ‘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 sensor ‘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 sensor 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 sensor 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 sensors.

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

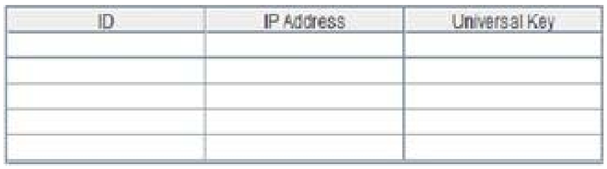
Decryption is the reverse process:

• 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 sensors 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 sensors detection and after detection)

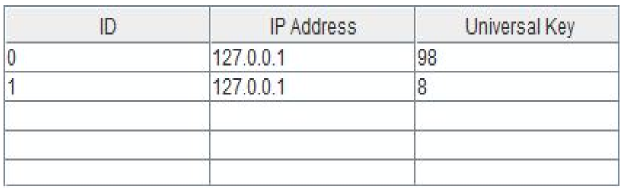


ID	IP Address	Universal Key

Figure3: KeySender table before detection

ID is the unique number given to each sensor, IP address is the logical address and universal is the symmetric key used for encryption and decryption

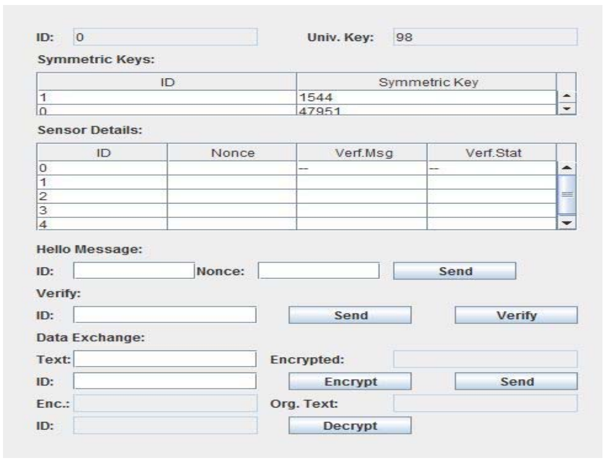
Key Sender Table after sensor detection:



ID	IP Address	Universal Key
0	127.0.0.1	98
1	127.0.0.1	8

Figure4: Key-sender table after clients/sensors are detected

Here we take 2 sensor 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:

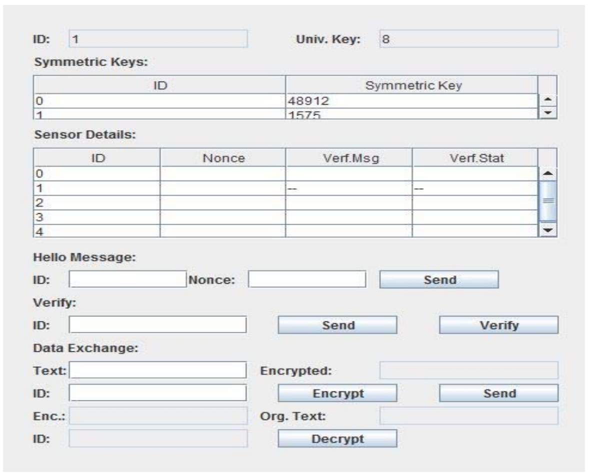
ID	Symmetric Key
1	1544
0	47951

Sensor Details:

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

Hello Message:
ID: Nonce:
Verify:
ID:
Data Exchange:
Text: Encrypted:
ID:
Enc.: Org. Text:
ID:

Figure5:Receiver0


ID: 1 Univ. Key: 8

Symmetric Keys:

ID	Symmetric Key
0	48912
1	1575

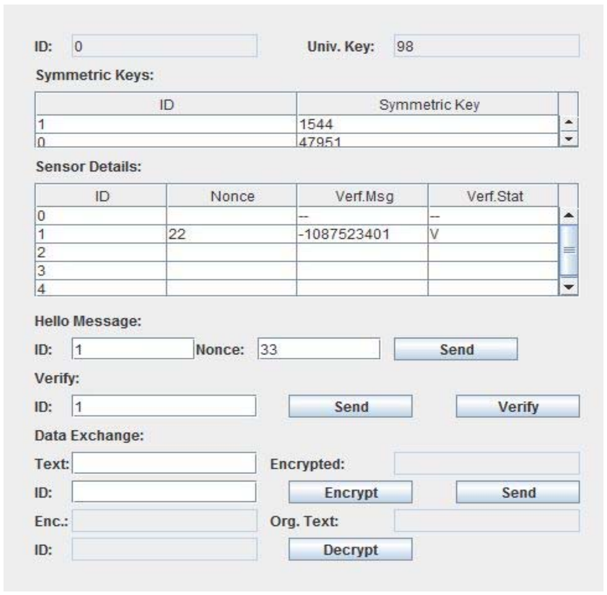
Sensor Details:

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

Hello Message:
ID: Nonce:
Verify:
ID:
Data Exchange:
Text: Encrypted:
ID:
Enc.: Org. Text:
ID:

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:

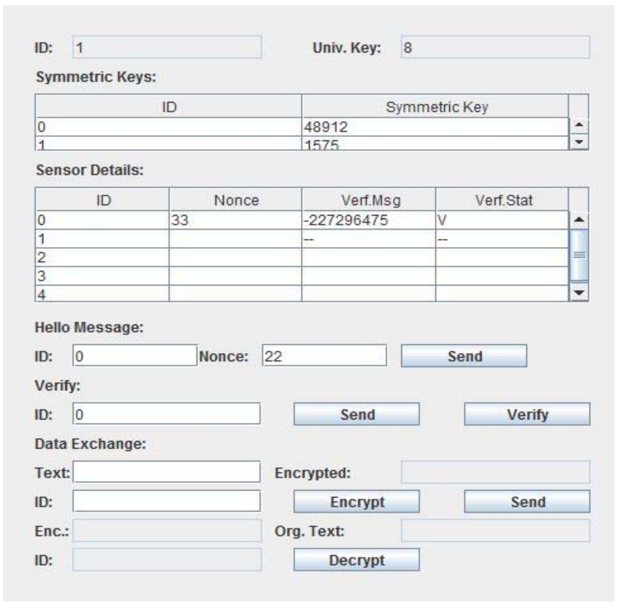
ID	Symmetric Key
1	1544
0	47951

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
Data Exchange:
Text: Encrypted:
ID:
Enc.: Org. Text:
ID:

Figure7:Reciever0 with keys updated


ID: 1 Univ. Key: 8

Symmetric Keys:

ID	Symmetric Key
0	48912
1	1575

Sensor Details:

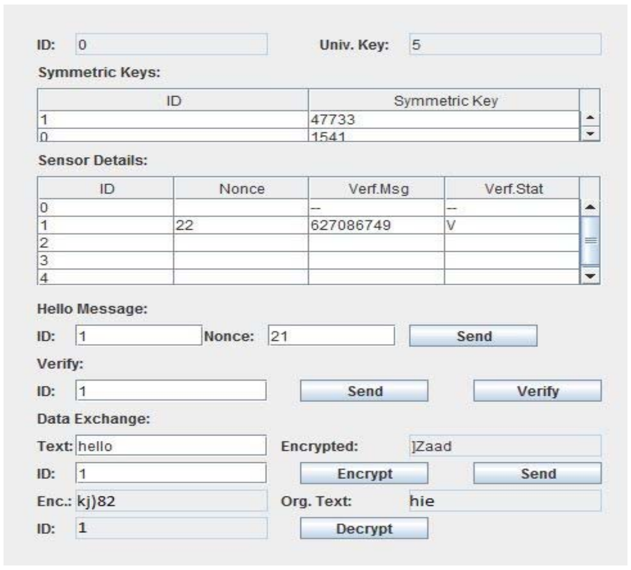
ID	Nonce	Verf.Msg	Verf.Stat
0	33	-227296475	V
1		--	--
2			
3			
4			

Hello Message:
ID: 0 Nonce: 22
Verify:
ID: 0
Data Exchange:
Text: Encrypted:
ID:
Enc.: Org. Text:
ID:

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

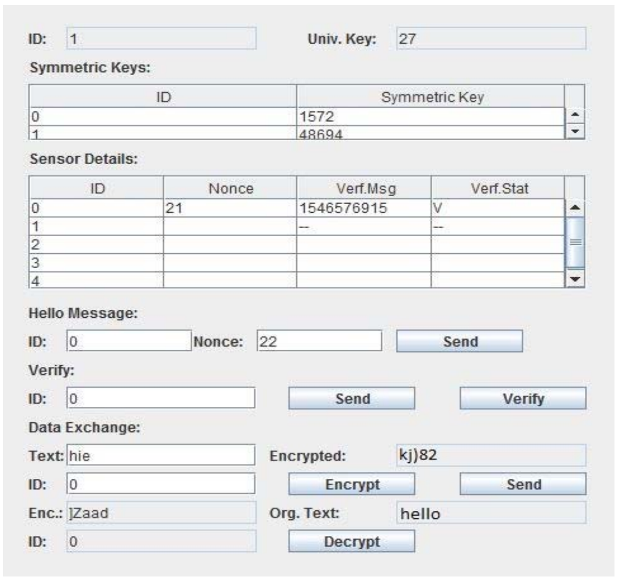
Sensor Details:

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	48694

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.:]Zaad 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

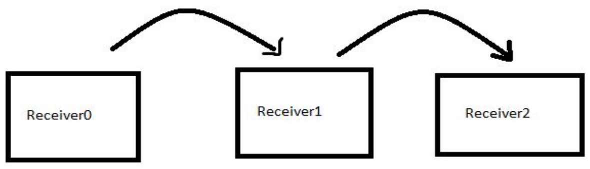


Figure9: nodes communication

REFERENCES

Communication within Sensor Networks by Using Key Distributor by ch.d.naiduijcsit 2014.

The remaining references are:

1. Taehwan Choi, H. B. Acharya and Mohamed D. Gouda, " The Best Keying Protocol ", December 2011 IEEE
2. http://en.wikipedia.org/wiki/Sensor -Sensors
3. Sensor Networks by Margaret Rouse.
4. http://en.wikipedia.org/wiki/Mobilewirelesssensornetwork-MWSN's
5. "Key Distribution Mechanisms for Wireless Sensor Networks" by Seyit A., C¸Amtepe and BulentYener
6. "Cryptography and Network Security", Fourth Edition by William Stallings
7. L. Gong and D. J. Wheeler, "A matrix key-distribution scheme," Journal of Cryptology, vol. 2, pp. 51–59, January 1990.
8. "Advanced Encryption Standard" by Douglas Selent
9. http://en.wikipedia.org/wiki/Advanced_Encryption_Standard#cite_note-fips-197-4.
10. "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.
11. Steganography Detection using Functional Link Artificial Neural Networks, International Journal of Computer Applications (0975 - 888), Volume 47 No.5 June 2012.
12. 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.