Implementation of Threshold Cryptography in MANETS

P. Swetha

Associate Professor, Computer Science & Engineering Department, JNTUHCEJ, Andhra Pradesh, India,

Abstract

Security is the most desirable feature in Mobile Ad hoc NETworks (MANET). In order to have secured communication between two parties, a secret key is used for encrypting and decrypting messages. The communication in a mobile network can be protected by ensuring that a secret key can be known only to the two communicating parties. The distribution of the secret key can be achieved by a Cryptographic technique called Threshold Cryptography (TC). In this technique, the secret key is divided into multiple shares, where these shares are distributed to the nodes participating in the communication in an infrastructure less network. In TC, another module Proactive Secret Sharing (PSS) is introduced, which allows the set of nodes to refresh the shares by generating a new set of shares without reconstructing the secret key. TC along with PSS provides essential security by not exposing the secret key.

Index Terms: MANET, Threshold Cryptography, Proactive Secret Sharing, Share refreshment, Lagrange's Interpolation.

1. Introduction

guidelines In wireless communication, a Mobile Adhoc NETwork (MANET) is an emerging area for new developments. Security in a MANET, is an important issue which is ruling the internet world today. A MANET is a network which provides wireless communication between heterogeneous devices with least or no infrastructure [1]. A MANET is a self-configuring network which dynamically adapts a temporary network topology for establishing a internetwork for communication between people and devices without any preexisting infrastructure. Each node participating in communication provides services such as message forwarding, authentication, routing information etc. for creating a temporary network. A route for forwarding data packets is established between arbitrary nodes for sending and receiving packets. The legitimate nodes must establish a secure communication channel to thwart against threats, eavesdrops or tampering of the data. Due to the unique characteristics of MANET such as dynamic topology, infrastructure less wireless network, it is mandatory to provide security [2]. Implementing a secure ad hoc network has become a challenging task because of the vulnerabilities and limited computational and storage capabilities. Hence, the basic security requirements of MANET are availability, authentication, integrity, confidentiality, authorization. With all these constraints, the most critical and complex issue is the distribution of a secret group key to the legitimate nodes in a secure fashion which is used to encrypt the data.

Considering the nature and challenges posed by the MANET and to provide security, Threshold Cryptography (TC) is employed. TC is a cryptographic technique, sharing secret among legitimate nodes [3]. In (n, t)TC, a secret key is divided into 'n' shares using a cryptographic operation and these shares are distributed to the nodes participating in communication. The secret can be reconstructed only when the threshold number 't' out of 'n' shares are combined together. The individual shares alone cannot serve the purpose of reconstructing the original secret key. A secret group key cannot be constructed with fewer than 't' shares.

The basic requirement is that, within the share transmission procedure, each share must not be disclosed. However there is a possibility for a malicious node to generate the secret key by stealing 't' or more shares from the participating nodes, within long span of time. In order to escape from the threats of exposing a secret key, a Proactive Secret Sharing (PSS) can be introduced. PSS plays an important role as the key management protocol using threshold cryptography. In PSS, each share is updated periodically, such that a malicious node cannot construct the secret key within the short time span [4] [5]. In non-proactive secret sharing, if the threshold number of shares are compromised during the lifetime of the secret, the secret is compromised. In PSS scheme, all shares are refreshed by generating a new set of shares for the same secret key from the old shares and then the old shares are discarded after the each share is refreshed. All shareholders must cooperate with the PSS procedure for the protocol consistency.

This paper presents TC implementation using PSS. The paper is organized as follows, section II discusses about the analysis of TC. Section III about the analysis of PSS protocol. Section IV about the simulation and results and finally section V concludes the paper with its future scope.

2. Threshold Cryptography

MANETs are susceptible to attacks because of the use of wireless links. The network confidentiality is violated when eavesdroppers are successful in accessing the secret information. The security services like availability, integrity, authentication and non-repudiation are violated when hackers try to attack the network to delete packets, or inject erroneous packets or impersonate a node. Compromised nodes launch attacks from within a network. The routings algorithms like on-demand and link-state algorithms fail to protect the data or sensitive routing information [6]. A centralized entity or a single node in a MANET is not trustworthy which could lead to vulnerabilities. For this, a security solution based on the distribution of trust is required. TC is used to distribute the trust to an aggregation of nodes.

In (n,t) TC scheme, a secret key is divided into ’n’ shares and shared among ‘n’ nodes using some cryptographic operation. Any node can collect ‘t’ threshold number of shares and can reconstruct the original key ’K’ [7]. On contrary, it is infeasible for at most ‘$t-1$’ nodes to construct the key ’K’ even by collusion. A ‘$t-1$’ degree polynomial is constructed with the constant secret key ‘K’ and random elements:

$$y = f(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + a_{t-3}x^{t-3} + \cdots + a_1 x + K$$

In this case, each of the ‘n’ shares is a pair of (x_i, y_i) of numbers such that $f(x_i) = y_i$, where $i \in \{1,2,3,\dots.n\}$, $x_i \neq 0$. Given ‘t’ shares, the secret key ‘K’ is computed using Lagrange’s Interpolation. Therefore, TC is one of the techniques suitable for MANETs in key sharing and distributing to multiple nodes because (1) it does not need any key infrastructure, (2) it works even in a busy network, where atleast ‘t’ nodes must reside in the same network. Hence, (n,t) TC scheme can be defined as a concrete key management system in a MANET environment with one secret group key and its ‘$t-1$’ degree polynomial. Here, ‘n’ shareholders must be able to perform the cryptographic operation for the distribution of the trust. If a new node wishes to join the secure group communication, it must collect ‘t’ or more shares from the share-holders to generate the secret group key ’K’. The new node has to be authenticated before the transmission of the shares from each share-holder. The distribution of the shares among the nodes is achieved through a secure link with the help of Private/Public keys [8] where the nodes are certified by Certificate Authority (CA) [9]. The new node has to construct the secret group key without the knowledge of pre-used polynomial. The secret group key and the polynomial have to be securely initialized.

The share-generation can be defined as a three -step process:

1. Pick random coefficients $a_{t-1}, a_{t-2}, \dots, a_2, a_1$.
2. Build a polynomial of degree $'t-1'$ such that

$$y = f(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + a_{t-3}x^{t-3} + \cdots + a_1 x + a_0$$

where a_0 is the secret key.

3. Each participant receives a unique share ‘x’ and its corresponding$f(x)$.

In order to generate the secret key, the $(x, f(x))$ pairs are collected from ‘t’ participants to rebuild the polynomial.

3. Proactive Secret Sharing

A countermeasure to mobile adversaries, who can reconstruct a secret key within long time span, is the proposal of proactive schemes. A Proactive TC uses share refreshment technique in which the participating nodes in collaboration must refresh their shares from the old shares. The newly refreshed shares constitute a new (n,t) TC. After refreshing, the old shares are discarded and the new shares are used. Now the adversary cannot combine old shares with the new shares to generate the secret key. Hence, the challenging task to the adversary is to compromise ‘$t+1$’ nodes between periodic refreshing.

3.1 Share Refreshment Algorithm

Without using any key infrastructure, TC gives a way to convey a shared key to a node which is suitable for secret sharing in MANETs. However, given ‘t’ shares in (n,t) TC, the secret can be found. The major shortcoming of TC is that, if a malicious node has stolen ‘t’ shares, within finite span of time, the original secret key can be generated. Therefore, share refreshment becomes necessary component, to refresh each share from the old ones without reconstructing the secret key [12]. For the share refreshment, each shareholder generates their own sub-shares and distributes these sub-shares to other participating nodes through a secure link, to refresh their own shares.

Algorithm Steps:

1. Let a secret key ‘K’ be divided into ‘n’ shares $(k_1, k_2, k_3, \dots k_n)$ with node ‘i’ having k_i and distributed to ‘n’ nodes.
2. Each node ‘i’ having k_i, generates their own subkeys $\{k_{i1}, k_{i2}, \dots k_{in}\}$
3. Every subkey k_{ij} $(j \in \{1,2,\dots,n\})$ is distributed to node j through secure link.
4. When node j gets the subkey $(k_{1j}, k_{2j}, k_{3j}, \dots k_{nj})$ it, refresh the old share to new shares as

$$K_j' = k_j + \sum_{i=1}^{n} k_{ij}$$

After generating the new share K_j', the old share k_j is discarded.

5. Now, the new shares are $(k_1', k_2', k_3', \dots, k_n')$ are an (n,t)sharing of the secret key K, because $\sum_{j=1}^{n} k_{ij} = 0$, for all $i \in \{1,2,\dots,n\}$

After the PSS procedure, each node holds the refreshed shares where the old shares become useless. In this case, the malicious node must collect atleast ‘t’ shares before the refreshment which becomes impossible.

The block diagram for Threshold cryptography using PSS is shown. It is clearly depicted that first a secret key K is randomly generated which is divided and distributed among nodes. Later, if all nodes agrees for share refreshment , PSS procedure is started. After successful completion of PSS procedure, a node has to collect ‘t’ shares and generate the secret key ‘K’.

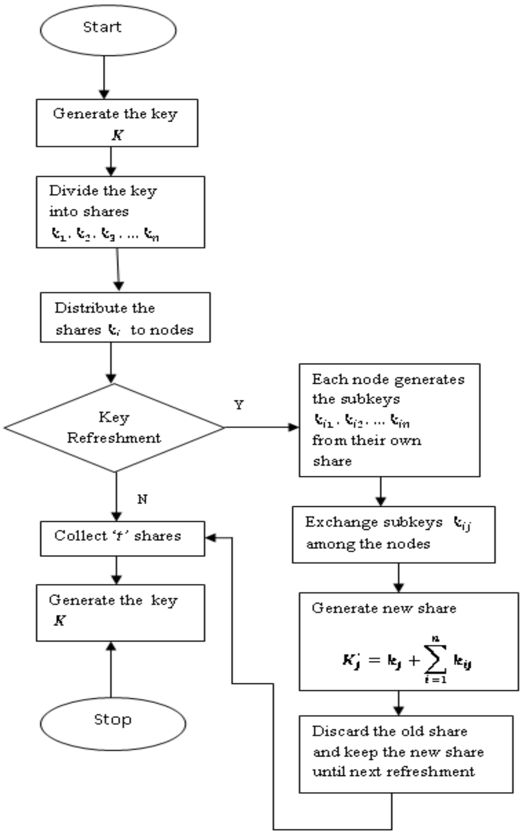

Start
Generate the key K
Divide the key into shares $t_1, t_2, t_3, \dots t_n$
Distribute the shares t_i to nodes
Key Refreshment
Y
N
Each node generates the subkeys $t_{i1}, t_{i2}, \dots t_{in}$ from their own share
Exchange subkeys t_{ij} among the nodes
Generate new share
$K_j' = k_j + \sum_{i=1}^{n} k_{ij}$
Discard the old share and keep the new share until next refreshment
Collect 't' shares
Generate the key K
Stop

Fig.3.1 : Flow Chart of Threshold Cryptography using PSS

The PSS procedure has to be initiated with proper timing coordination i.e., all shareholders must synchronize with the PSS procedure. Otherwise, if a node 'A' starts PSS and other node 'B' does not start PSS procedure , then node 'A' receives old shares of 'B' and 'B' receives new shares of 'A' i.e., they exchange inconsistent shares, there by the secret key cannot be generated. Hence, it leads to protocol inconsistency. Therefore, it is assumed that all share-holders start the PSS procedure with proper timing coordination.

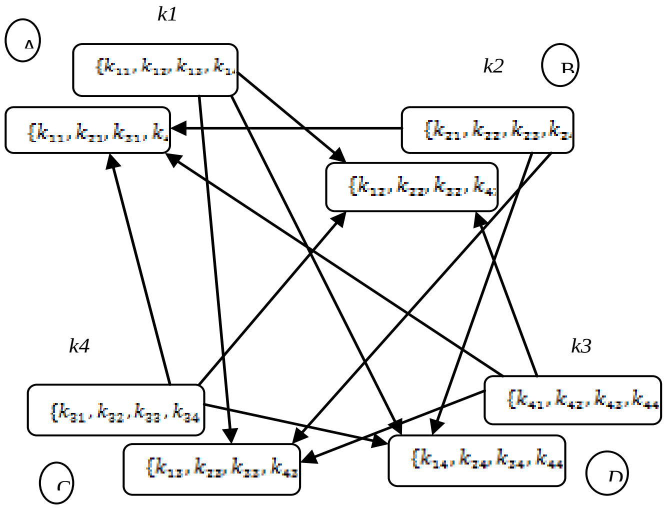

k1
A
$\{k_{11}, k_{12}, k_{13}, k_{14}\}$
k2
B
$\{k_{11}, k_{21}, k_{31}, k_{41}\}$
$\{k_{21}, k_{22}, k_{23}, k_{24}\}$
$\{k_{12}, k_{22}, k_{32}, k_{42}\}$
k4
k3
$\{k_{31}, k_{32}, k_{33}, k_{34}\}$
$\{k_{41}, k_{42}, k_{43}, k_{44}\}$
C
$\{k_{13}, k_{23}, k_{33}, k_{43}\}$
$\{k_{14}, k_{24}, k_{34}, k_{44}\}$
D

Fig -3.2: Share Refreshment Procedure

Fig -3.2 shows the share refreshment procedure. Assume a secret key $'K'$ is divided into four shares k_1, k_2, k_3, k_4, which are distributed to the nodes A,B,C,D respectively. Each of these shares in turn are divided into sub-shares by their respective nodes. . Node A generates the subkeys $\{k_{11}, k_{12}, k_{13}, k_{14}\}$ from k_1. Similarly nodes B,C and D also generates the subkeys $\{k_{21}, k_{22}, k_{23}, k_{24}\}$, $\{k_{31}, k_{32}, k_{33}, k_{34}\}$, $\{k_{41}, k_{42}, k_{43}, k_{44}\}$ from k_2, k_3, k_4, respectively. Every subkey k_{ij} is distributed to node j as given in step3 of the algorithm. When node j receives the sub-share $\{k_{1j}, k_{2j}, k_{3j}, \dots k_{nj}\}$ it, refresh the old share to new shares as

$$K_j' = k_j + \sum_{i=1}^{n} k_{ij}$$

After producing the new shares, old shares become obsolete. Now these new shares are used for distribution among other nodes.

4. Implementation

In this section, TC along with PSS implementation is discussed. The evaluation was conducted using Network Simulator-2[13]. The code for creating a wireless network topology is done in TCL language. The nodes were created using wireless links and provide routing algorithm to route the data to the corresponding destination. The size of the simulation area is 700x700 with at most 100 nodes. The encryption is done by using DES symmetric encryption along with AODV protocol[14].

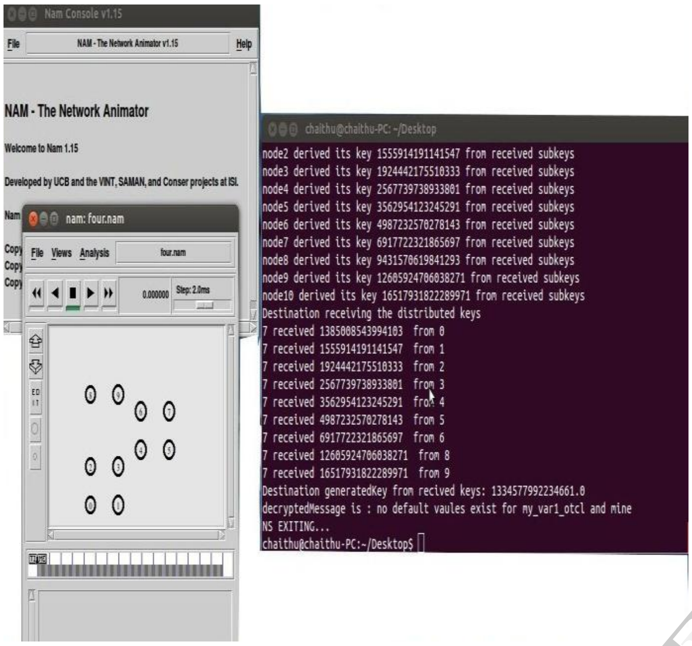
The following parameters are assumed given in Table:1:

Simulator	NS-2
Number of Nodes	10
Threshold Value	5
Routing protocol	AODV
Wireless simulation area	700X700
Channel type	Wireless
Mac type	MAC/802.11
Simulation duration	600 Units
Traffic	User Data

Table-1: List of parameters

Threshold cryptography with Proactive secret sharing is run in ns-2 and the simulation is shown in Network Animator (NAM) [15][16][17]. After execution the simulation will generate key shares and they are distributed to other nodes for key refreshment.

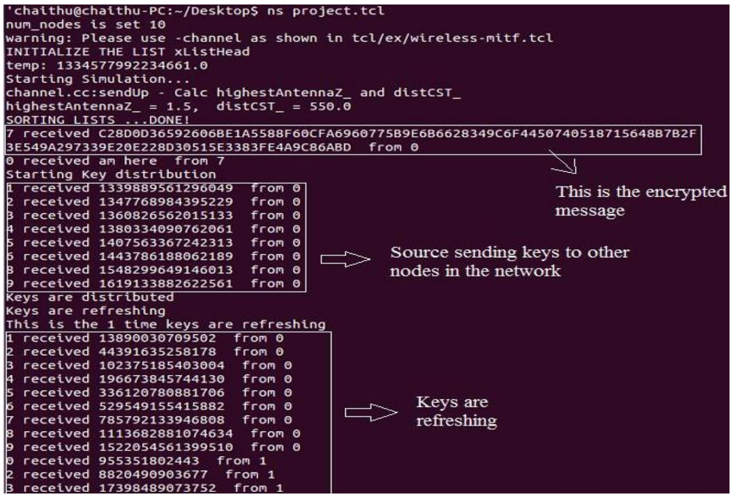
We get the following interface.


Nam Console v1.15
File
NAM - The Network Animator v1.15
Help
NAM - The Network Animator
Welcome to Nam 1.15
Developed by UCB and the VINT, SAMAN, and Conser projects at ISI.
nam: four.nam
File
Views
Analysis
four.nam
0.000000
Step: 2.0ms
0
0
0
0
0
0
0
0
0
0
chaithu@chaithu-PC: ~/Desktop

```
node2 derived its key 1555914191141547 from received subkeys
node3 derived its key 1924442175510333 from received subkeys
node4 derived its key 2567739738933801 from received subkeys
node5 derived its key 3562954123245291 from received subkeys
node6 derived its key 4987232570278143 from received subkeys
node7 derived its key 6917722321865697 from received subkeys
node8 derived its key 9431570619841293 from received subkeys
node9 derived its key 12605924706038271 from received subkeys
node10 derived its key 16517931822289971 from received subkeys
Destination receiving the distributed keys
7 received 1385008543994103  from 0
7 received 1555914191141547  from 1
7 received 1924442175510333  from 2
7 received 2567739738933801  from 3
7 received 3562954123245291  from 4
7 received 4987232570278143  from 5
7 received 6917722321865697  from 6
7 received 12605924706038271  from 8
7 received 16517931822289971  from 9
Destination generatedKey from recived keys: 1334577992234661.0
decryptedMessage is : no default vaules exist for my_var1_otcl and mine
NS EXITING...
chaithu@chaithu-PC:~/Desktop$
```

Fig-4.1: NAM output showing mobile nodes and the terminal showing key exchange between nodes

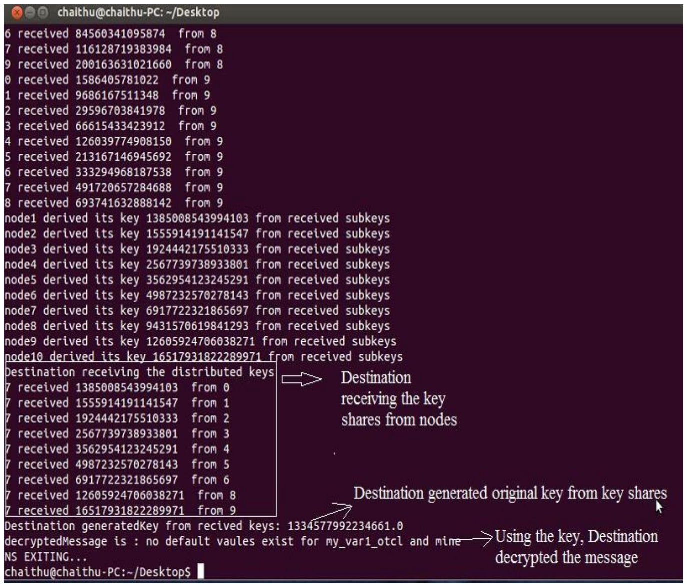
In **Fig 4.1**, the Network Animator displays 10 mobile nodes. The terminal window displays each node receiving subkeys from other nodes.


This is the encrypted message
Source sending keys to other nodes in the network
Keys are refreshing

```
'chaithu@chaithu-PC:~/Desktop$ ns project.tcl
num_nodes is set 10
warning: Please use -channel as shown in tcl/ex/wireless-mitf.tcl
INITIALIZE THE LIST xListHead
temp: 1334577992234661.0
Starting Simulation...
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
7 received C28D0D36592606BE1A5588F60CFA6960775B9E6B6628349C6F4450740518715648B7B2F
3E549A297339E20E228D30515E3383FE4A9C86ABD  from 0
0 received am here  from 7
Starting Key distribution
1 received 1339889561296049  from 0
2 received 1347768984395229  from 0
3 received 1360826562015133  from 0
4 received 1380334090762061  from 0
5 received 1407563367242313  from 0
6 received 1443786188062189  from 0
8 received 1548299649146013  from 0
9 received 1619133882622561  from 0
Keys are distributed
Keys are refreshing
This is the 1 time keys are refreshing
1 received 13890030709502  from 0
2 received 44391635258178  from 0
3 received 102375185403004  from 0
4 received 196673845744130  from 0
5 received 336120780881706  from 0
6 received 529549155415882  from 0
7 received 785792133946808  from 0
8 received 1113682881074634  from 0
9 received 1522054561399510  from 0
0 received 955351802443  from 1
2 received 8820490903677  from 1
3 received 17398489073752  from 1
```

Fig-4.2: Key refreshment

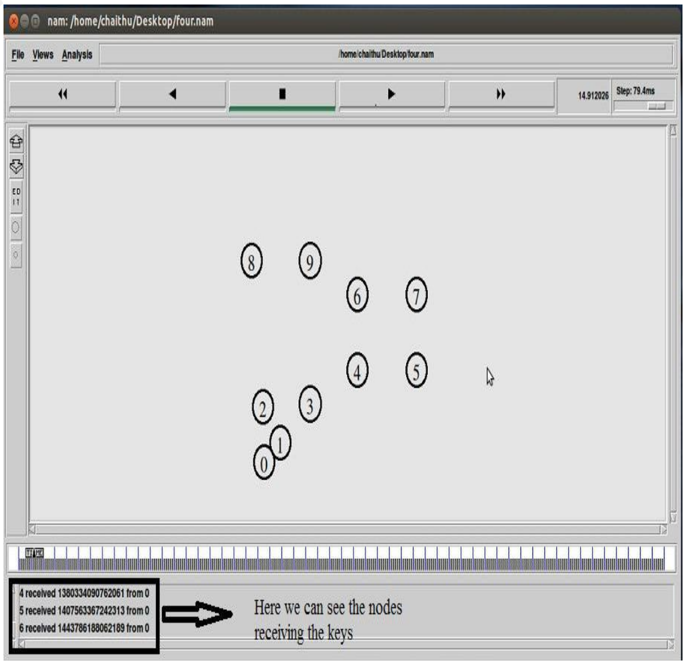
In the **Fig 4.2**, it shows the encrypted text transmitted from source to the destination. Source sending its subkeys to the other nodes. Later, each node refreshes their respective keys.


chaithu@chaithu-PC: ~/Desktop
Destination receiving the key shares from nodes
Destination generated original key from key shares
Using the key, Destination decrypted the message

```
6 received 84560341095874  from 8
7 received 116128719383984  from 8
9 received 200163631021660  from 8
0 received 1586405781022  from 9
1 received 9686167511348  from 9
2 received 29596703841978  from 9
3 received 66615433423912  from 9
4 received 126039774908150  from 9
5 received 213167146945692  from 9
6 received 333294968187538  from 9
7 received 491720657284688  from 9
8 received 693741632888142  from 9
node1 derived its key 1385008543994103 from received subkeys
node2 derived its key 1555914191141547 from received subkeys
node3 derived its key 1924442175510333 from received subkeys
node4 derived its key 2567739738933801 from received subkeys
node5 derived its key 3562954123245291 from received subkeys
node6 derived its key 4987232570278143 from received subkeys
node7 derived its key 6917722321865697 from received subkeys
node8 derived its key 9431570619841293 from received subkeys
node9 derived its key 12605924706038271 from received subkeys
node10 derived its key 16517931822289971 from received subkeys
Destination receiving the distributed keys
7 received 1385008543994103  from 0
7 received 1555914191141547  from 1
7 received 1924442175510333  from 2
7 received 2567739738933801  from 3
7 received 3562954123245291  from 4
7 received 4987232570278143  from 5
7 received 6917722321865697  from 6
7 received 12605924706038271  from 8
7 received 16517931822289971  from 9
Destination generatedKey from recived keys: 1334577992234661.0
decryptedMessage is : no default vaules exist for my_var1_otcl and mine
NS EXITING...
chaithu@chaithu-PC:~/Desktop$
```

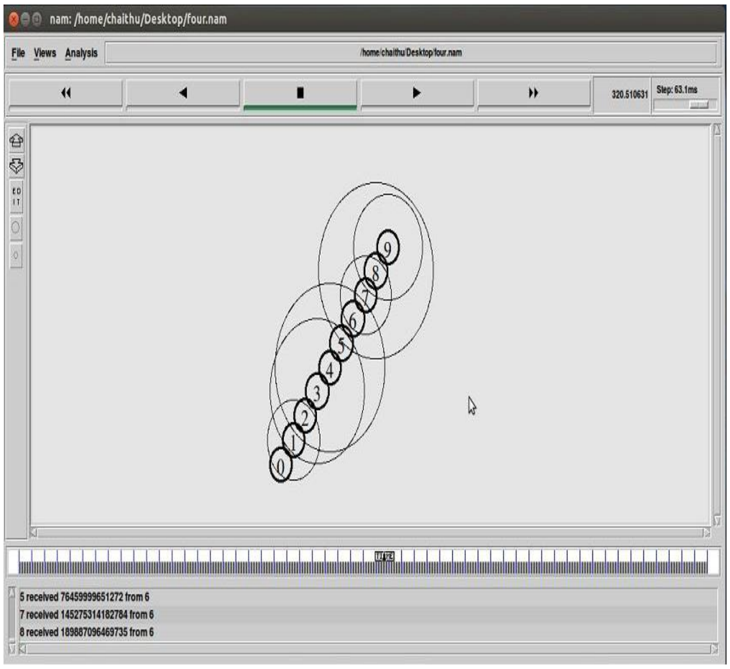
Fig-4.3: Destination received the key shares

Fig 4.3 displays that destination receives subkeys from other nodes and constructs the original key. This generated key will be used for decrypting the message received from the source. **Fig 4.4** and **Fig 4.5** displays the same i.e., key exchange and key refreshment process in the Network animator.


nam: /home/chaithu/Desktop/four.nam
File
Views
Analysis
/home/chaithu/Desktop/four.nam
14.912026
Step: 79.4ms
8
9
6
7
4
5
2
3
1
0
Here we can see the nodes receiving the keys

```
4 received 1380334090762061 from 0
5 received 1407563367242313 from 0
6 received 1443786188062189 from 0
```

Fig-4.4: Nodes receiving the keys


nam: /home/chaithu/Desktop/four.nam
File
Views
Analysis
/home/chaithu/Desktop/four.nam
320.510631
Step: 63.1ms
EDIT
0
1
2
3
4
5
6
7
8
9
TIME

```
5 received 76459999651272 from 6
7 received 145275314182784 from 6
8 received 189887096469735 from 6
```

Fig-4.5: Keys are refreshing and sending their shares to corresponding nodes

The simulation can be run for 50 to 100 nodes. But as the number of nodes increases the , the key generation and refreshment slows down since it increases the time complexity for key exchanges among all the nodes.

5. Conclusion

This paper focuses on analyzing Threshold Cryptography which protects from the direct exposure of secret key which is used for encryption and decryption of packets. Proactive secret sharing acts as an add-on to TC by refreshing the sub shares periodically by which a malicious node cannot construct the original secret key from the old sub shares. This is an efficient technique which helps to enforce the security in a mobile network. TC along with PSS can defend against most of the security attacks because the secret key itself is partitioned into multiple subshares. This technique works efficiently only when all the nodes are synchronized to start up PSS procedure. If not, a refreshed subshare cannot be generated. Developing novel synchronization procedures for the PSS protocol consistency would be the future work. Additional work can be done by appending a hash value to the secret key for the integrity and investigate the network performance as the mobile nodes increases.

6. References

[1]. A. Mishra and K. M. Nadkarni, "Security in wireless adhoc networks - A Survey"', in The Handbook of Ad HocWireless Networks, M. Ilyas, Ed. Boca Raton: CRC Press, 2002.
[2]. G.S. Mamatha, Dr. S. C. Sharma, "A Highly Secured Approach against Attacks in MANETS" , International Journal of Computer Theory and Engineering, Vol. 2, No. 5, October, 2010 ,1793-8201
[3]. A. Shamir, "How to share a secret," Communication of the ACM, vol.22, 1979.
[4]. A. Herzberg, J. Stanislaw, H. Krawczyk, M. Yung, "Proactive Secret Sharing or: How to Cope with Perpetual Leakage," Proc. 15th Annual International Cryptology Conference on Advances in Cryptology, 1995, pp. 339-352.
[5]. A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In D. Coppersmith, editor, Advances in Cryptology—Crypto'95, the 15th Annual International Cryptology Conference, Santa Barbara, CA USA, August 27–31, 1995, Proceedings, volume963 of Lecture Notes in Computer Science. Springer, 1995.
[6]. Neha Gupta, Manish Shrivastava, "Securing Routing Protocol by Distributed Key Management and Threshold Cryptography in Mobile Ad hoc Network" , International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970) Volume-3 Number-1 Issue-9 March-2013.
[7]. Marianne A. Azer, Magdy S. El-Soudani, "Threshold Cryptography and Authentication in Ad Hoc Networks -Survey and Challenges" , Second International Conference on Systems and Networks Communications (ICSNC 2007),0-7695-2938-0/07 IEEE , 2007.
[8]. Y. Kitada, A. Watanabe, K. Takemori, and I. Sasase, "On demand distributed public key management for wireless ad hoc network"' in IEEE Pacific Rim Conference on Communication, Computers and Signal Processing, 2005.
[9]. H.Mohri, I. Yasuda, Y. Takata, and H. Seki, "Certificate chain discovery in web of trust for ad hoc networks," in proceeding of the 21st International Conference on Advanced Information Networking and Applications workshop, IEEE Computer Society, vol.2,pp,2007.
[10]. T-79.159 Cryptography and Data Security, 24.03.2004 Lecture 9: Secret Sharing, Threshold Cryptography,
MPC, Helger Lipmaa , pg no.11-20.
[11].echidna.maths.usyd.edu.au/~kohel/tch/MATH3024/.../lectures_11.pdf
[12]. Hitoshi Asaeda, Musfiq Rahman, Yoshihiro Toyama, "Structuring Proactive Secret Sharing in Mobile Ad -hoc Networks ",0-7803-9410-0/06,2006 IEEE.
[13]. T. Issariyakul and E. Hossain, Introduction to Network Simulator NS2, Springer 2008.
[14]. C. Perkins, E. Belding-Royer and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC3561, July 2003.
[15]. ns-2 Home page : http://www.isi.edu/nsnam/ns/
[16].ns-2Tutorial:
http://www.isi.edu/nsnam/ns/tutorial/nsindex.html
[17]. Tutorial for the Network Simulator "ns" by Marc Greis.