Secure Transmission Services in Wireless Networks using Key Trees

Geddam.Kiran Kumar
Sri Vasavi Engineering College, Pedatadepalli,
Tadepalligudem. W.G.Dt., A.P. India.

Prof K .V. Mutyalu
Sri Vasavi Engineering College, Pedatadepalli,
Tadepalligudem. W.G.Dt., A.P. India.

Abstract

In contrast with conventional networks, mobile networks usually do not provide on-line access to trusted authorities or to centralized servers for secure message transmissions and exhibit problems like frequent problems due to link-node failures and pricey issues. For these reasons, existing solutions for secure message transmission services in regular networks require on-line trusted authorities or certificate repositories that are not well suited for securing mobile networks. To reduce computation overhead the core system is deployed at a Server and accessed via mobile nodes. In this paper, we propose a fully self-organized public-key management system that allows users to generate their key message pairs and cipher/decipher them(using RSA encryption), to issue access, to perform authentication regardless of the network partitions and number of users using Key Tree management schemes.

Keywords - Wireless broadcast, key management, access control, key hierarchy, secure group communication, key distribution.

I. INTRODUCTION

The demand of wireless networks (WNs) is growing exponentially. It has turned out that the sensor networks can be widely applied in the areas of healthcare, environment monitoring, and the military. One of the surveys on WNs points out that, in the near future, wireless sensor networks will be an integral part of our lives, more so than the present-day personal computer [1].

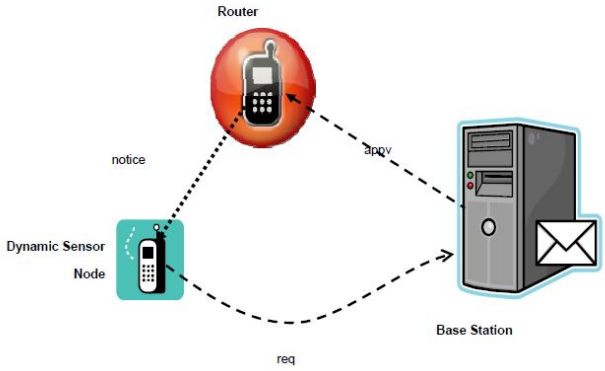
A sensor node has low capability in terms of power, computation, storage and communication. A wireless sensor network is composed of a large number of wireless sensor nodes and multi-hop communication is desired in WSNs. As a result, security in wireless sensor networks has six challenges to overcome: (i) the wireless nature of communication, (ii) resource limitations of sensor nodes, (iii) very large and dense WSNs, (iv) lack of fixed infrastructure, (v) unknown network topology prior to deployment, (vi) high risk of physical attacks on unattended sensors [2]. Basic Protocol Due to the limited storage of sensor nodes, the pre-shared key-pair is not always available between the roaming node and its new neighbors in the circumstance of a dynamic node roaming within large

WSNs (e.g., in hospitals and nuclear power plants). Therefore it requires an efficient and scalable protocol to establish and update the keys among nodes for secure communications. Figure 2 shows the basic architecture and message flow of our protocol for authentication and key establishment in dynamic WSNs. When a dynamic sensor node moves to a new area and wants to attach to a router or a cluster head in this area, it first sends a request message to the base station (refer to Figure).

req={Src=SN, Dst= BS, RT||R0||MAC(KBN, SN||RT||R0)}

where Src and Dst denote the source and destination address of a message respectively. SN, BS and RT are identifiers for sensor node, base station and router, respectively. R0 denotes a random number generated by the sensor node. MAC indicates the message authentication code algorithm with a key and KBN is the shared secret key between the base station and the sensor node.

Figure 2. The basic architecture and message flow of our protocol.



After receiving the req message, the base station will check its revocation list whether the sensor node has been revoked. If the sensor node is acceptable, then the base station verifies the MAC message. If the result is positive, the base station will generate a session key KNR for the roaming sensor node and the router (or cluster head). KNR = H(KBN, SN||R0||R1)

where H is a keyed one-way hash function, and R1 is the random number selected by the base station. The base station then sends an approval message appv with the session key to

the router: appv = {Src=BS, Dst=RT, E(KBT, SN||R0||R1||KNR)} (3) where E is an encryption algorithm, and KBT is the shared secret key between the base station and the router.After receiving the appv message, the router decrypts the payload and extracts the session key KNR, and then sends a notice to the sensor node. notice = {Src=RT, Dst=SN, R0||R1|| MAC(KNR, RT||SN|| R0||R1)} (4) Upon getting the notice message, the sensor node extracts the random numbers R0 and R1. After checking if the received random number R0 is equal to the original R0, the sensor node recalculates the session keyKNR = H(KBN, SN||R0||R1) and then verifies the MAC value. If the result is positive, the sensor node will use the session key for the communication with this router afterwards. In practice, the router could be any sensor node that the dynamic sensor node wants to connect to.

In computer communication services, the basic data unit is a data item. The data items are grouped in to programs, and the user specifies the required program to perform the specific task. A user may subscribe to one or more programs and the set of subscribed programs are called the user's subscription. The users can communicate through the Internet or uplink channels to specify the programs that they are interested in receiving. Wireless data broadcast services have mainly focused on performance issues such as reducing data access latency and conserving the battery power of mobile devices but the critical security requirements of this type of broadcast services have not yet been addressed as the service providers need to ensure backward and forward secrecy, with respect to membership dynamics.

In the wireless broadcast environment, any user can monitor the broadcast channel and record the broadcast data. If the data is not encrypted, the content is open to the public, and anyone can access the data . In addition, a user may only subscribe to a few programs. If data in other programs are not encrypted, the user can obtain data beyond his subscription privilege. Subscription is a messaging pattern in which the senders of messages do not program the messages to be sent directly to specific receivers or subscribers. The basic wireless data broadcasting system is shown in figure-2.

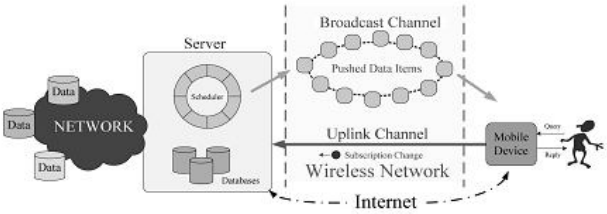


Fig.1.A wireless data broadcast system.

The published messages are characterized into classes and the access control should been forced by encrypting the data in a proper ways such that only subscribing users can access the broadcast data, and subscribing users can only access the data to which they subscribe[9].

II. RELATED WORK

Kimetal [6] proposed a combination of key tree and Diffie-Hellman key exchange to provide a simple fault-tolerant key agreement for collaborative groups. The working [2] reduces the number of rekey messages, while [9] and [2] improve the reliability of rekey management. Balanced and unbalanced key trees are discussed in [5] and[2]. Periodic group rekeying is studied in[7] and [8] to reduce the rekey cost for groups with frequent joins and leaves. Issues on how a key tree is maintained and how encrypted keys are efficiently placed in multicast rekey packets are studied in [8] and [2] Moreover, the performance of LKH is thoroughly studied [3],[8]. In broadcast Encryption, there are some key management schemes in the literature for multicast and broadcast services. Briscoe [2] used arbitrarily revealed key sequences to do scalable multicast key management without any over head on joins leaves. Wool[8] proposed two schemes that insert an index head in to packets for decryption. Lubyand Staddon[7] proposed a scheme for yield in gmaximal resilience again starbitrary coalition so for on privileged users. However, the size(entropy) of its broadcast key messages large, at least a zero-message scheme [7], [8] which does not require the broadcast server to disseminate any message in order to generate a common key.

Many users subscribe to multiple programs simultaneously which requires separation to avoid overlapping. Since multiple programs are allowed to share the same set of keys, a critical issue is how to manage shared keys efficiently and securely. In many circumstances, when a user subscribes to new programs or unsubscribe to some programs, a large portion of keys that the user will hold in his new subscription can be reused without compromising security.

The following schemes were used for user management and program management.

1. Logic Key Hierarchy (LKH) secure key management for wireless broadcast is closely related to secure group key management in networking. The data encryption key (DEK) of the program and each represents an individual key (IDK) of a user that is only shared between the system and the user. Other keys in the tree, namely key distribution keys (KDKs), When a user joins or leaves the group, the server needs to change and broadcast the corresponding new keys, and this operation is called rekey, and the broadcast message of new keys is called rekey message. In our system, data and rekey messages are broadcast in the same broadcast channel to the users.

2. Broadcast encryption techniques

There are some other key management schemes in the literature for multicast and broadcast services. Used arbitrarily

revealed key sequences to do scalable multicast key management without any overhead on joins/leaves. Proposed two schemes that insert an index head into packets for decryption. However, both of them require pre-planned subscription, which contradicts the fact that in pervasive computing and air data access a user may change subscriptions at any moment. Compared with LKH-based approaches, key management schemes in broadcast encryption are less flexible regarding possible subscriptions.

3. Rekey Operations:
To issue new keys upon a user event, the main task is to identify the keys that need to be changed. We use two types of paths in the key forest to represent the to-be-changed keys. When a user leaves a tree, we say, a leave path is formed, which consists of keys that the user will no longer use. When a user joins a tree, we say, an enroll path is formed, which consists of keys that the user will use in the future. Similarly, when a user shifts from one tree to another, a leave path and an enroll path are formed. In KTR, a complete path starts from the leaf node and ends at the multiple DEKs of the subscribed programs that share the tree. To broadcast new keys, the server should first compose rekey packets.

III. PROPOSED ARCHITECTURE

First, the proposed scheme takes advantage of the facts in broadcast services. The proposed concept has more than a couple of programsthat (instead of many like in existing system) needs to be managed like Key Trees for user management and RSA encryption policy for secure messaging. The following schemes were used for user management and program management.

1. Logic Key Hierarchy (LKH):
We confine the use of logic key hierarchy only for user management and authentication purposes.

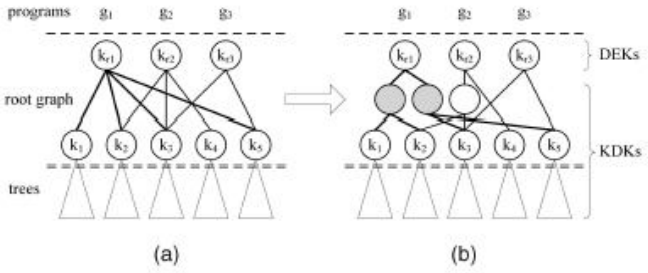
2. Unicast encryption techniques

Unicast requires only one client to attend to unlike Broadcast and Multicast. So we use RSA key management scheme for addressing the issue of secure message transmission service.

3. Rekey Operations:

Since we need use once instances of the service we don't require rekey operations. In the broadcast channels, the main mechanism for data dissemination is that users can recover lost or missed data items. The uplink channels, which have limited bandwidth, are reserved for occasional uses to dynamically change subscriptions. the ever-growing popularity of smart mobile devices ,along with the rapid advent of wireless technology, there has been an increasing interest in wireless data services among both industrial and academic communities in recent years .Among various approaches, broadcast allow save efficient usage of the scarce wireless bandwidth, because it allows simultaneous access by an arbitrary number of mobile clients. Wireless data broadcast services have been available as commercial products for many years.

Key Forest
In order to address scalability and flexibility in key management, an intuitive solution is to use a key tree for each program.LKH[7] is used as the basis of our scheme., but when the user u1 subscribes to two programs simultaneously, he needs to manage two sets of keys in both trees, which is not very efficient , hence SKT is proposed to reduce this cost in key management. We let the two programs share the same sub key tree, so that users subscribing to both programs only need to manage the keys in the gray triangle. The advantage of SKT is that any user subscribing to both g1 and g2 only needs to manage one set of keys for both programs. Moreover, when a user joins or leaves a tree shared by multiple programs the encryption and communication cost for rekey operations can be significantly less than conventional LKH approaches fig 2. In order to ensure that a user will not pay for subscribed programs multiple times; the key forest obviously should have the following properties, which are guaranteed in any directed acyclic graph which is an abstract representation of a set of objects where some pairs of the objects are connected by links.



The user activities of joining or leaving or shifting among trees instead of joining or quitting or changing among programs is the mapping between the tree-oriented operations and the corresponding program-oriented user events. When a user us shifts from tr4 to tr6 and when us was in tr4, us subscribed g1 and g2, he shifts to tr6, he subscribes g1, g2, and g3, the shift, in fact, means that the user adds g3 into his current subscription. This has the ability to change a lock so that a different key may operate it.

IV. EXPERIMENTAL RESULTS

This is the process of obtaining, analyzing, and recording information about the relative worth of the system. It is the analysis of successes and failures and suitability for further improvement. The performance of KTR at the server side and the client side, respectively are analyzed.

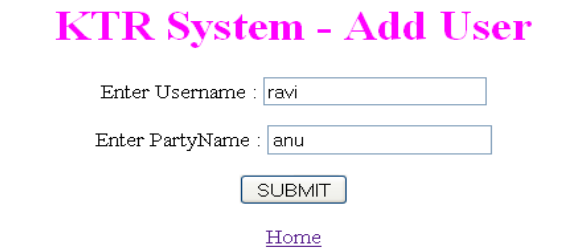
SAMPLE IMPLEMENTATION CODE:

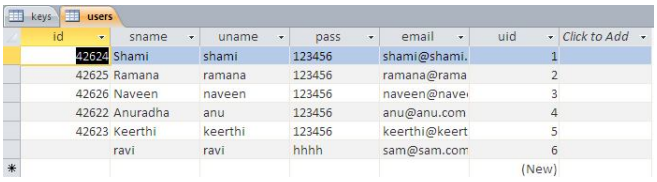
```
package test.ktr;

import java.io.InputStream;
import java.util.ArrayList;

import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.telephony.gsm.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class Messaging extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);
   setContentView(R.layout.messaging);

   Button basic = (Button) findViewById(R.id.basic);
   //Button salt = (Button) findViewById(R.id.salt);
   Button send = (Button) findViewById(R.id.send);
   Button read = (Button) findViewById(R.id.read);
   Button decrypt = (Button) findViewById(R.id.decrypt);

   final EditText phone1    =   (EditText)
findViewById(R.id.txtPhoneNo);
   final EditText message  =   (EditText)
findViewById(R.id.message);
   //final EditText password = (EditText)
findViewById(R.id.password);
   final EditText changedmessage1 = (EditText)
findViewById(R.id.changedmsg);
   final EditText decryptmsg =   (EditText)
findViewById(R.id.decryptmsg);


        final TextView msgtype =  (TextView)
findViewById(R.id.msgtype);

   decryptmsg.setOnClickListener(new
View.OnClickListener() {

        @Override
        public void onClick(View arg0) {
          try
          {
            decryptmsg.setText("");
          }
          catch(Exception e)
          {
            Toast.makeText(Messaging.this,"Unable to
Clear:"+e.toString(), Toast.LENGTH_LONG).show();
          }

        }

   });

   decrypt.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View arg0) {
          try
          {
            // default salt 123456
            String salt    = "123456";
            String passwordEnc =
decryptmsg.getText().toString().trim();
            Toast.makeText(Messaging.this,"Encrypted
Message:"+passwordEnc, Toast.LENGTH_LONG).show();
            sendToServer();
            String passwordDec =
 SimpleCrypto.decrypt(salt,passwordEnc);
            Toast.makeText(Messaging.this,"Decrypted
Message From Server:"+passwordDec,
Toast.LENGTH_LONG).show();
            //msgtype.setText("Decrypted Message");
            decryptmsg.setText("");
            decryptmsg.setText(passwordDec);
          }
          catch(Exception e)
          {
            Toast.makeText(Messaging.this,"Unable to
Decrypt:"+e.toString(), Toast.LENGTH_LONG).show();
          }

        }

   });
```

Server Side Analysis

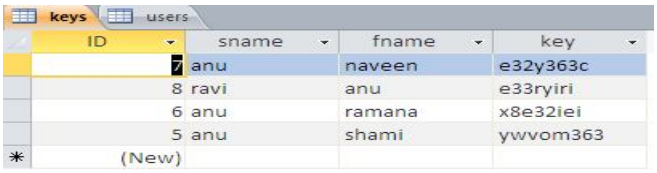
This is the analysis done on a computer program running as a service, to serve the needs or requests of other programs which may or may not be running on the same computer. Since a server is generally abundant in energy and memory, its computation capacity becomes the main factor that affects the performance of the whole system. If the processing time for each event is large, this would delay a user's request. We measured the management cost of a server by two metrics

which is the total number of keys to be managed and the number of keys to be inspected and updated per rekey event.





id	sname	uname	pass	email	uid	Click to Add
42624	Shami	shami	123456	shami@shami.	1	
42625	Ramana	ramana	123456	ramana@rama	2	
42626	Naveen	naveen	123456	naveen@nave	3	
42622	Anuradha	anu	123456	anu@anu.com	4	
42623	Keerthi	keerthi	123456	keerthi@keert	5	
	ravi	ravi	hhhh	sam@sam.com	6	
					(New)	

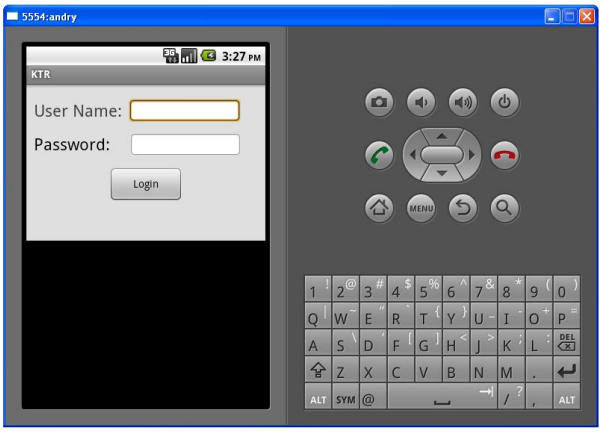


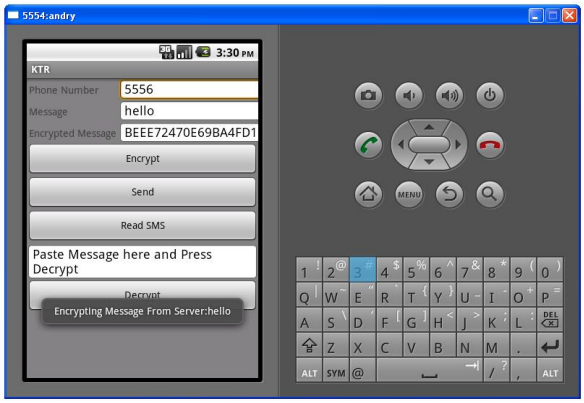
ID	sname	fname	key
7	anu	naveen	e32y363c
8	ravi	anu	e33ryiri
6	anu	ramana	x8e32iei
5	anu	shami	ywvom363
(New)			

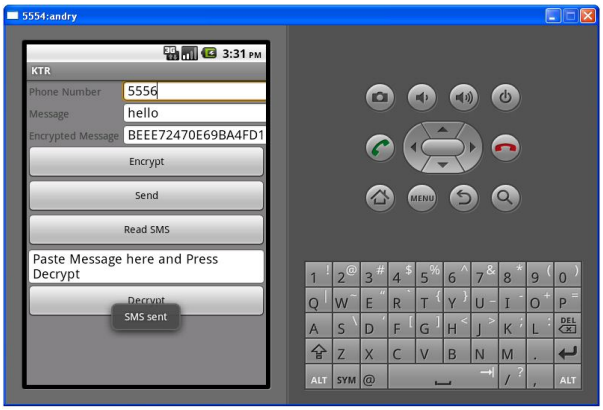
Client Side Analysis

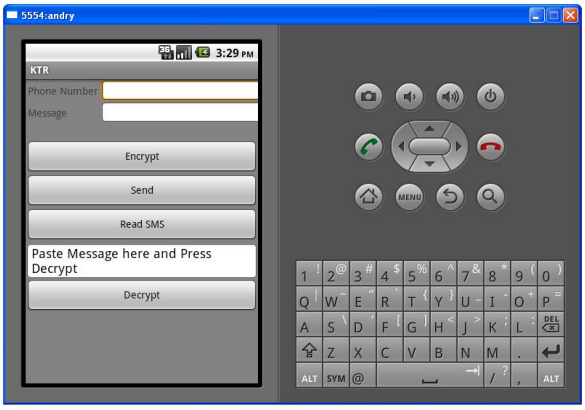
There are some main performance metrics that are to be measured. They are , the average rekey message size per event, average number of decryption per event per user, and maximum number of keys to be stored which can well capture the overhead of KTR on resource-limited mobile devices in terms of communication, storage, power consumption, and computation. Based onthe metrics, we can infer other metrics that are more directly related to the mobile devices. We can obtain metrics such as the communication overhead in the rekey messages.

```
package test.ktr;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.gsm.SmsMessage;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class SmsReceiver extends BroadcastReceiver
{
   @Override
   public void onReceive(Context context, Intent intent)
   {
      //---get the SMS message passed in---
      Bundle bundle = intent.getExtras();
      SmsMessage[] msgs = null;
      String str = "";
      if (bundle != null)
      {
         //---retrieve the SMS message received---
         Object[] pdus = (Object[]) bundle.get("pdus");
         msgs = new SmsMessage[pdus.length];
         for (int i=0; i<msgs.length; i++){
            msgs[i] =
SmsMessage.createFromPdu((byte[])pdus[i]);
            str += "SMS from " +
msgs[i].getOriginatingAddress();
            str += " :";
            str += msgs[i].getMessageBody().toString();
            str += "\n";
         }
         //---display the new SMS message---
         Toast.makeText(context,"Incoming Message : "+str,
Toast.LENGTH_SHORT).show();
      }
   }
}
```









V. CONCLUSION AND FUTURE WORK

The issues of key management in support of secure wireless broadcast services. We proposed the KTR as a scalable, efficient, and secure key management approach in the broadcast system. We used the key forest to exploit the overlapping nature between users and programs in broadcast services. KTR lets multiple programs share a single tree so that the users subscribing these programs can hold fewer keys. In addition, we proposed a novel shared key management approach to further reduce rekey cost by identifying the minimum set of keys that must be changed to ensure broadcast security. This approach is also applicable to other LKH-based approaches to reduce the rekey cost as in KTR.

REFERENCES

[1] Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless sensor networks: a survey.
Comput. Netw. **2002**, *38*, 393-422.

[2] Camtepe, S.A.; Yener, B. *Key Distribution Mechanisms for Wireless Sensor Networks: a Survey*; Technical Report TR-05-07; Department of Computer Science, Rensselaer Polytechnic Institute: Troy, NY, USA, March 2005.

[3] J. Snoeyink, S. Suri, and G. Varghese, "A Lower Bound forMulticast Key Distribution," Proc. IEEE INFOCOM '01, vol. 1.
[4] LeinHarn and ChangluLin "Authenticated Group Key Transfer Protocol Based On SecretSharing " IEEE Transactions On Computers, Vol. 59, No. 6, June 2010

[5] Xingyu Li and H,Vicky Zhao "An Efficient Key Management Scheme For Live Streaming" IEEE Communications Society,IEEE "Globecom" 2009.

[6] Ohtake,Goichiro Hanaoka, and Kazuto Ogawa "An Efficient Provider Authentication ForBidirectional Broadcasting Service" IEEE Transactions On Broadcasting,January2009.

[7] Xiaoguang Niu , Yanmin Zhu , Li Cui , Lionel M. Ni "FKM: A Fingerprint-based KeyManagement Protocol for SoC-based Sensor Networks"IEEE Communications Society, IEEE 2009.

[8] Xukai Zou Elisa Berti Yuan-Shun Dai "A Practical and Flexible Key Management Mechanism For Trusted Collaborative Computing" IEEE Communications Society, IEEE "Globecom" 2009.

[9] Reusable Key Management Technique for Secured Wireless Computer Communication S.Afrin Banu *Anna University Tirunelveli.* Nagercoil-3, K.K Dist Tamilnadu,India. afrinbanu1986@gmail.com.