Volume 5, Issue 6, June 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Development and Validation of a NS-2 Protocol at Network Layer

Naval Kishor Lodhi
Institute of Information Technology,
Gwalior, India

Abstract-Network Simulator version 2 (NS-2) is adiscrete event, object oriented simulation tool to simulate and analyze dynamic nature of communication networks. NS-2 is an open source to develop new protocols and functions. It provides supports for Transmission Control Protocol/Internet Protocols (TCP/IP) and many standard routing protocols for wire and wireless networks. In NS-2 many protocols have been implemented at various layers of TCP/IP protocol suite to provide different functionalities but none provide the security functions. This paper addresses this issue by adding new security module or protocol in NS-2. Security module helps in key sharing as well as implementation of encryption/decryption functions with the help of deffiehellman algorithm, which is a symmetric key algorithm and Transpositional cipher algorithm. This paper describes the features of security module in details, discuss the algorithms used, simulation process and implementation of security module on the basis of NS- 2. Result of implementation shows that data transfer from source to destination is more secure.

Keywords- Network Simulator (NS-2), Encryption/Decryption, Key Sharing, Deffie-hellman Algorithm, Symmetric Key Cryptography, Security at Network Layer, Transpositional Cipher.

I. INTRODUCTION

Network Simulator (NS-2) is an open source system that is developed using C++ and Tool Control Language (TCL). The version of NS-2 used in this paper is 2.35 [1, 2]. Within this version, most of the standard protocols supported. Protocols can be found from media access layer such as Carrier Sense Multiple Access with Collision Detection (CSMA/CD) up to application protocols such as File Transfer Protocol (FTP), Telecommunication Networks (TELNET) and Hyper Text Transfer Protocol (HTTP). For routing protocols, there are unicast and multicast routing protocols for wire networks and Dynamic Source Routing (DSR), Destination Sequenced Distance Vector (DSDV), Ad Hoc On Demand Distance Vector (AODV) for wireless ad-hoc networks. Most of these protocols were developed by researches and adopted into standard version of NS-2 [3, 4]. This paper illustrates a way to add security module by which key sharing functions can be done in NS-2. The new protocol, defining a new packet format, is specific to a wired network at network layer. The new protocol is represented by a class packet_sec derived from built in classes in NS-2. Within new derived class, add encryption/decryption functions for data field in the data packet. In addition to above, the implementation of key sharing functions to ensure the confidentiality of data packet during transmission. Data is considered as plain text. The encryption/decryption and key exchanging algorithms are transpositional cipher and deffie-hellman respectively. The programming languages are C++, TCL and AWK [5]. The remaining portion of this paper is organized as follows: section 2 present the related work. Section 3 describes the security algorithms used for key exchange and encryption/decryption. Section 4 will present the proposed work. Section 5 and 6 describe the implementation of wired network and experimental studies respectively. In section 7, the analysis of result and performance are measured and in the last section 8, conclude the paper and future scope.

II. RELATED WORK

In NS-2 very less work has been done in the field of security on data. Author JIANG Hong *et al. [6]* of East China Normal University have proposed their work in this direction. They enhance the security of Ethernet by a group based on MAC key selection protocol (GKSP) for large Ethernet networks. They provided security at data link layer i.e. security has hop to hop jurisdiction. Recently researches are focusing on to propose a new protocol for wired and wireless networks and some are trying to enhance or optimize the previously available protocols. For wired network, application layer protocols like TELNET, FTP, Cluster Based Routing (CBR), HTTP etc., transport layer protocols like TCP, User Datagram Protocol (UDP) etc., networks layer protocols like Internet Protocol (IP), Ping etc. and data link layer protocols like CSMA/CD, proposed by researches. Later these protocols were incorporated in NS-2[7, 8].
This paper developed a security protocol to solve the problem for both key sharing and encryption/decryption of data at network layer. The programming platform used, NS-2 based on Red Hat Linux.

III. SECURITY ALGORITHMS USED

Two security algorithms are used for key exchange and encryption/decryption of data. They are describe below-

3.1. Deffie-Hellman Algorithm

In this algorithm, a secret key is generated and exchanged between the sender and receiver. A secret key is generated because it is needed for secure transmission of data [5, 9].

Let *q* and *a* be Global Parameters (G.P). During the key generation process, initially sender A will assume the value of and keep this value hidden. Now, sender A will calculate the value of intermediate key, according to the given formula:

$$Ya = a^{Xa} Mod\ q \quad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Similarly, receiver B assumes the value of variable and calculates intermediate key .

$$Y_b = a^{Xb} Mod\ q \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Now, sender and receiver exchange the value of and , and calculate the secret key accordingly:

$$k = Y_b{}^{Xa} Mod\ q \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

$$k = Y_a{}^{Xb} Mod\ q \quad \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Now, equation (3) and (4) generate the secret key, which will use for secure transmission of data over insecure channel.

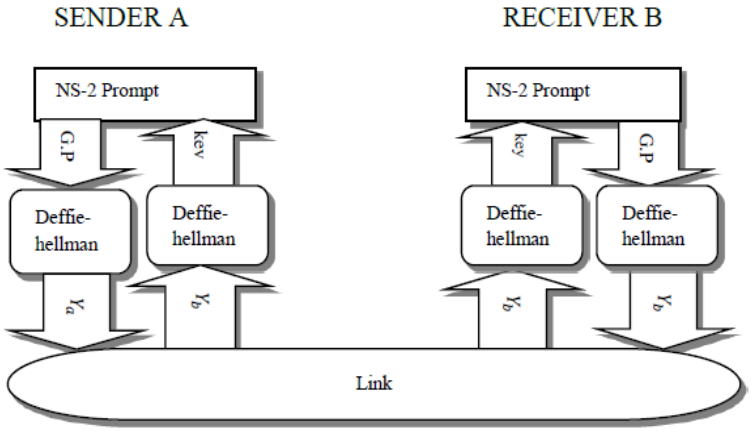

Figure 1: Logical design of key exchange process.

3.2. Encryption Algorithm

A columnar transpositional cipher, also known as row column transpose cipher, is very easy cipher to perform by hand. The message is written in rows of fixed length by creating columns. These columns are then rearranged and the message is again read column by column. Both the length of rows and subsequent arrangement of columns are defined either by a keyword or numerical key. In a regular columnar transpositional cipher, any square spaces are filled with nulls and in an irregular columnar transpositional cipher, the spaces are left blank [5].

Finally the message is read in columns, in order to specify by the key. To decipher a columnartranspositional cipher, the recipient must work with the column length divided by the message length and key length. Remove all spaces before decipher the text and ignore new lines and not taken into consideration.

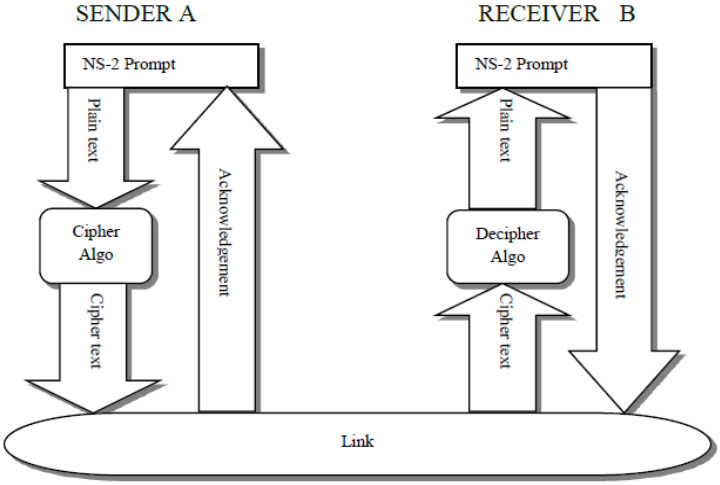

Figure 2: Logical design of the encryption/decryption function.

IV. PROPOSED WORK

The self-defined security protocol class is packet_sec. This class implements key sharing features as well as encryption/decryption functions. The code is given below:

```
Class packet sec: public Agent
{
  Public:
  //Accept data from the upper protocol and process.
  Void recv(Packet *p, Handler*);
  //Command processing function.
  Int command(int argc, const char* const* argv);
  Private:
  //Sequence number to check retransmission packet.
  Uint32t myseq;
};
```

Figure 3: Command and receive function of agent class.

This security protocol maintains a sequence number *seq* for each node. When data source sends a packet, then *seq* will add 1and *seq* will be added to the packet header. Through this packet header, receiving node can re-arrange the coming packets. In this security protocol source file such as packet_sec.h/.cc should be created. *Recv* function commands are inherited from agent class, shown in figure 3.

/packet sec can specify the packet_sec class in C++. Agent/packet sec should use to edit the TCL code and define the security protocol. Here, agent/packet sec demonstrate the inheritance relationship that packet_sec is inherited from agent.

4.1. Implementation of key exchange algorithm.

Algorithm 1: Sender A sends intermediate key to receiver B.

```
//Input: Global parameters- q and a.
Variables- Xa and Xb.
//sender side for sending key Ya for secret key generation
//at the sender side
if(strcmp(argv[1]), "exchange") ==0)
{
//create a new packet
Packet*pkt = allocpkt();
//access the packet header for the new packet:
hdrpacket*hdr = hdrpacketsec::access(pkt);
//set the „ret" field to 2, so the receiving node
//knows that it has to generate a packet with
//Yb value
hdr->ret=2;
hdr->seq=seq++;
//store the current time in the „send_time" field
hdr->send_time= scheduler::instance().clock();
//copy key to be sent to header
Xa= 18;
hdr->key= deffie_hellman(Xa);
printf("secret key send");
//send the packet
send(pkt, 0);
}
//end of sender side
//Output: function deffie_hellman() generates Ya and send it to the receiver B.
```

Algorithm-1 shows that how to send the value of by assigning to the key field of packet header. Initially, create a packet using the standard function allocpkt() then access the packet header by creating the packet header *hdr* of type hdrpacketsec. The packet sending time value is assigned to the send_time field of packet header to avoid the third party attack on deffie-hellman algorithm. This time value work as a time stamp. For calculating the value of *Ya*, sender A pass the value of to the function named deffie_hellman(). This function returns the value to the key field of packet header

Algorithm 2: Receiver receives intermediate key and generate a secret key k.

```
//Input: Global parameters- q and a.
Variables- Xa and Yb.
//Receiver side code for secret key generation and sending //the value of Yb to sender for calculating
secret key at
//sender side
//Access the IP header for the received packet:
hdr ip* hdrip = hdr ip::access(pkt);
//Access the security packet header for the received packet
hdrpacketsec*hdr = hdrpacketsec::access(pkt);
if(hdr->ret == 2)
{
//First save the old packet‟s send time
Double stime = hdr->seq;
//deffir_hellman1() function for secret key
//Calculation
Xb = 22;
Kb = deffie_hellman(Xb, hdr->key);
Packet::free(pkt);
//create a new packet
//creating packet for sending key value Yb at
//sender side
Packet*pktret = allocpkt();
//Access the header for the new packet:
hdrpacketsec*hdrret =
hdrpacketsec::access(pktret);
//Set the „ret‟ field to 3
hdrret->ret = 3;
hdrret->key = deffie_hellman(Xb);
send(pktret, 0);
}
//end of receiver side.
//Output: The function deffie_hellman() generates the secret key k and sebd it to the sender.
```

Algorithm-2 shows, how receiver B generates secret key k after receiving the value of how to calculate and send it to sender A. receiver B receives packet, removes the IP header from packet then access the security packet header and check the value of send_time field for validity of the packet. For secret key generation, receiver B pass the value of and to the deffie_hellman() function. It will generate a secret key k.

Receiver B generates by passing the value of variable to deffie_hellman() function and assign the value to the key field of the packet header and send the packet to the sender A.

Algorithm 3: Sender receives intermediate key and generates secret key k.

```
//Input: Global parameters- q and a.
Variables- Yb and Xa.
//sender receives Yband then calculate secret key used for
// further communication
if(hdr->ret == 3)
ka = deffie_hellman1(Xa, hdr->key);
packet::free(pkt);
//Output: function deffie_hellman() use the value of Xa
and generates a secret key.
```

Algorithm 3 shows, sender A receives the value of and pass this value with to deffie_hellman() function. This function will retureturn a secret key for secure communication.

4.2. Implementation of encryption/decryption function

Algorithm 4: Sender's encryption using transpositional cipher.

```
//Input: variables- ret,seq and send_time.
//Encryption using transposition cipher.
if(strcmp(argv[1], “send”) == 0)
{
//Create a new packet
Packet*pkt = allocpkt();
//Access the security packet header for the new
//packet:
```



```
hdrpacketsec* hdr = hdrpacketsec::access(pkt);
//set the „ret‟ field to 0, the receiving node knows
//that it has to generate an ack packet.
hdr->ret = 0;
hdr->seq = seq++;
//store the current time in the „send_time‟ field
hdr->send_time = scheduler::instance().clock();
//copied data to be send to header.
strcpy(hdr->data, argv[2]);
//--------------encrypt the data----------------
Encryption(hdr->data);
//send the packetsend(pkt, 0);
//return TCL_OK, so the calling function knows that the command has been processed.
return(TCL_OK);
}
//Output: Encryption function receive the encrypted data,
which is send to the receiver.
```

Algorithm 5: Receiver's decryption and acknowledgement sending.

```
//Input: variables- original_data[], send_time, rcv_seq, ack
and pkt.
//Packet received and decrypted.
if(hdr->ret == 0)
{
//Send an „Ack‟ and save the old packet‟s send
//time
Double stime = hdr->send_time;
//------------decrypt encrypt packet------------------
Char original_data[128];
//copy the data of original packet
strcpy(original_data, hdr->data);
int rcv seq = hdr->seq;
//show the encrypted data and decrypt it
decryption(original_data);
packet::free(pkt);
//create a new packet
Packet* pktret = allocpkt();
//Access the header of new packet:
hdrpacketsec* hdr2 =
hdrpacketsec::access(pktret);
//Set the „ret‟ field to 1, so the receiver would not
// send the another acknowledge.
hdrret->ret = 1;
//set the send_time field to the correct value
hdrret->send_time = stime;
hdrret->rcv_time = Schedulae::instance().clock();
hdrret->seq = rcv_seq;
//Send the packet back to the originator
send(pktret, 0);
}
//Output: Obtained the original datdata and sent the acknowledgement to the sender.
```

4.3. Changes made in TCL file.

TCL is a scripting language by which network components can be easily created. Here, an environment of six nodes, out of them security is attached on four nodes. They are connected to each other and the sequence of connections is: node0 is connected to node5 and node1 is connected to node4. Then the command is given from the TCL file for key exchange sending the data. The recv function, which access the value of variables display it on terminal after the successful execution. TCL pseudo code for key exchange data encryption/decryption is given below:

```
//TCL file pseudo code for exchange of data.
Setp0 [new Agent/Packet sec]
$ns attach-agent $n0 $p0
$p0 set class 1
set p1 [new Agent/Packet sec]
```



```
$ns attach-agent $n1 $p1
$p1 set class 1
set p2 [new Agent/Packet sec]
$ns attach-agent $n4 $p2
$p2 set class 2
set p3 [new Agent/Packet sec]
$ns attach-agent $n5 $p3
$p3 set class 2
#connect the two agent
$ns connect $p0 $p3
$ns connect $p1 $p2
$ns connect $p3 $p0 // not required
$ns connect $p2 $p1 // not required
#schedule events
for { set i 1} {$i!2} {incr i} {
set result [expr $i/2]
$ns at $result "$p0 exchange deffie"
$ns at [expr $result + 0.02] "$p1 exchange deffie"
$ns at [expr $result + 0.04] "$p0 send " it is along message"
$ns at [expr $result + 0.06] "$p1 send it is shortness"
$ns at [expr $result + 0.08] "$p2 send examine3"
$ns at [expr $result + 0.10] "$p3 send examine4"
}
#Define a „recv‟ function for the class
# „Agent/Packet_sec‟
Agent/Packet_sec instproc recvi {from rtt}
{
$self instvar node
Puts "node[$node id] received secret key from
$from trip_time $rtt ms"
}
```

This will generate a Network Animator (NAM) file, having six nodes from node0 to node5. Node0 and node1 will exchange the keys with node5 and node4 respectively.

V. IMPLEMENTATION OF WIRED NETWORK

In NS-2 the objective can be achieved by two ways:

i. When the built-in network components can not meet the purpose of simulation, it to create a new component or modify the existing components [10]. In other words, NS-2 should be extended by adding new Otcl class [11, 12].

ii. If it needs to simulate the basic mobile IPv4, then the components which are already available can directly be used. The establishment of special components can be composed of the required specific application, agents, links, routing and node model. These components are required to be tested and compiled, as to ensure whether they are correctly used [13]. After the compilation of simulation, the trace file should be analyzed to obtain valuable information network animator (NAM) can be used to monitor the network simulation process. The analysis results from modification can help to decide whether it needs to change the configuration topology.

Marc Greis‟s tutorial [3] shows how to add a new packet protocol into NS-2. Some steps are given below:

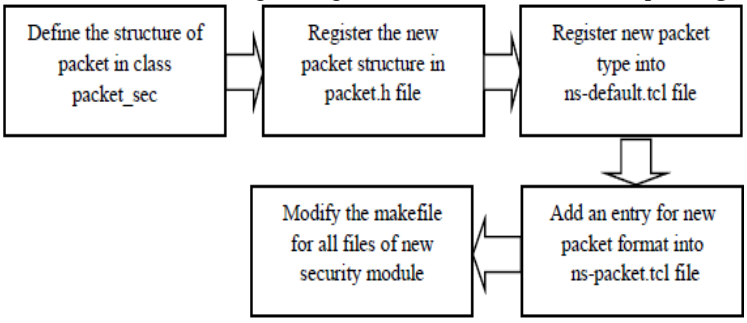

Figure 4: Registering the security module files.

i. The new packet class folder is created and defines the structure of packet in self-defined class packet_sec.

ii. Then register the new packet structure in packet.h file.

iii. At the TCL layer, the new packet must be declared by adding the name and default packet size value to the ns-default.tcl file.

iv. Now, make an entry for the new packet in the ns-packet.tcl file.

v. Finally, the make file has to be modified so that the new class can be compiled. After recompile the NS-2, the new packet can be used for simulation.

VI. EXPERIMENTAL STUDIES

Environment development requirements are:

i. Personal computer with RedHat operating system.
ii. NS-2 version 2.35
iii. C/C++ editor
iv. Gcc compiler

Environment can be generated using network animator (NAM):

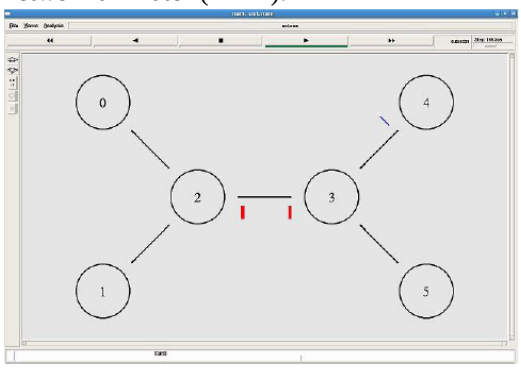

Figure 5: Environment generation using NAM.

In figure 5, an environment is generated after execution of NAM. It is having six nodes, which are connected to one another using full duplex link of 5 Mbps. Security agents are connected to n0, n1, n4 and n5 nodes. The communication takes place between nodes n0 to n5 and n1 to n4 through the link between n2 to n3. A 100 size of queue is in between nodes n2 and n3, it is a droptail type queue. The execution time is set for one minute, after the execution it will generate a trace file. The trace file will be processed using text processing language like Awk and get the desired output.

VII. RESULT AND PERFORMANCE ANALYSIS

After the compilation of test steps, the performance of the security protocol can be analyzed through the experimental data statistics

```
//-------------------------Key Generation----------------------//
node1 received Ya key from node0 with trip-time 10.2 ms
node0 received Yb key from node1 with trip-time 20.4 ms
Secret key generated; key value = 7

 //---------------------Message send by node0-----------------//
node0 message sent itisalongmessageIcan
Entered into encryption function: Ka = 7
After encryption contend: p{pzhsvunt1zzhn1Pjhu
Message sent
//-----------------Message received by node1-----------------//
node1 received packet from node0 with trip-time 10.2 ms
contend: p{pzhsvuntlzzhnlPjhu
decrypted itisalongmessageIcan
acknowledgement sent
//-----------------Message sent by node1----------------------//
node1 Message sent Itisashotermess
Entered into encryption function: Ka = 7
After encryption contend: P{pzhzovi{lytlzz
Message sent
//-----------------Message received by node0-----------------//
node0 received packet from node1 with trip-time 10.2 ms
contend: P{pzhzov{lytlzz
decrypted Itisashortmess
acknowledgement sent

//---------------------node1 receive ack-----------------------//
node1 received packet from node0 with trip-time 20.4 ms
contend: Message_Accepted.
```

Figure 6: Result after execution of security protocol

Figure 6, represents the packet transmission state using security protocol. In figure 5 and 6, the data is transferred in the form of unicast. When one node sends data packet, it will unicast that packet directly to the connected node. The node accepts the packet, check its destination address and forward it to other node in the form of unicast again. This process continues until the data packet reaches its destination

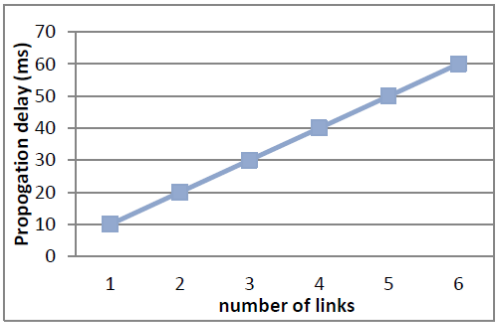

Figure 7: Propagation Delay (ms) with respect to links.

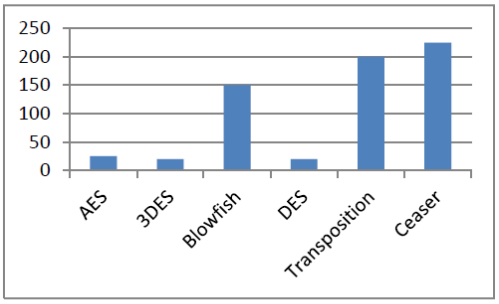

Figure 8: Throughput of security algorithms (mb/sec)

Figure 7 shows, the propagation delay with respect to links. Figure 8 shows, the throughput of different security algorithms (mb/sec). These algorithms are used for secure transmission of data. Security is measured in terms of generating public and private keys for encryption/decryption of data.

VIII. CONCLUSION AND FUTURE SCOPE

Simulations of wired network with two different protocols (key sharing and encryption/decryption) are presented with its result. Network Animator (NAM) is used to display the process of simulation. AWK is used for post processing of trace file and make comparison. On the basis of above a number of different topologies and data flow are alternated for simulation experiment and the results are nearly the same. Now, it can be said that security module is successfully deployed in NS-2, which is very useful for various applications like data security. Various applications can be analyzed after imposing the security on the data packets in terms of security overheads, packet loss, bandwidth required, throughput etc. In this module, deffie-hellman algorithm, which is a symmetric key exchanging algorithm and transpositional algorithm as an encryption algorithm are used. Transpositional cipher algorithm can be replaced with any symmetric key algorithm.

In later version, security module can be extended to support public key cryptography. This module is specific to wired networks but in future wireless networks will include and analyze the various applications after imposing security on data.

REFERENCES

[1] Qun, Z.A.; Wang Jun, "Application of NS2 in Education of Computer Net- works" Advanced Computer Theory and Engineering, 2008. ICACTE 08. Digital Object Identifier: 10.1109/ICACTE.2008.89, Publication Year: 2008, Page(s): 368 - 372.

[2] Ishak, Z.; Din, N.M.; Jamaludin, M.Z., "IPQit: An internet simulation kit based on NS2" Telecommunications and Malaysia International Con- ference on Communications, 2007. ICT-MICC 2007. Digital Object Identifier: 10.1109/ICTMICC.2007.4448686, Publication Year: 2007, Page(s): 489 - 493.

[3] Marc Greis, "Tutorial for the Network Simulator „ns"" Link: http://www.isi.edu/nsnam/ns/tutorial/.

[4] Tuteja, Asma; Gujral, Rajneesh; Thalia, Sunil, "Comparative Perfor- mance Analysis of DSDV, AODV and DSR Routing Protocols in MANET Using NS2" Advances in Computer Engineering (ACE), 2010. Digital Object Identifier: 10.1109/ACE.2010.16, Publication Year: 2010, Page(s): 330 - 333.

[5] William Stallings, "Cryptography and Network Security" Book, Fourth Edition, Publication Year: 2006, ISBN - 978-81-7758-774-6.

[6] Shumin Xu; Yatao Yang, "Protocols simulation and performance analysis in wireless network based on NS2" Multimedia Technology (ICMT), 2011. Digital Object Identifier: 10.1109/ICMT.2011.6003076, Publication Year: 2011, Page(s): 638 – 641.

[7] Guanghui Li; Jianming Chen, "The research of routing algorithms based on NS2 in mobile ad hoc networks" Software Engineering and Service Science (ICSESS), 2011. Digital Object Identifier: 10.1109/IC-SESS.2011.5982468, Publication Year: 2011, Page(s): 826 - 829.

[8] Guanghui Li; Jianming Chen, "The research of routing algorithms based on NS2 in mobile ad hoc networks" Software Engineering and Service Science (ICSESS), 2011. Digital Object Identifier: 10.1109/IC-SESS.2011.5982468, Publication Year: 2011, Page(s): 826 - 829.

[9] Lei Fan; Xu, C.X.; Li, J.H., "Deniable authentication protocol based on Deffie-Hellman algorithm" Electronics Letters Volume: 38 , Issue: 14, Digital Object Identifier: 10.1049/el:20020502, Publication Year: 2002, Page(s): 705 - 706.

[10] Tuteja, Asma; Gujral, Rajneesh; Thalia, Sunil, "Comparative Perfor- mance Analysis of DSDV, AODV and DSR Routing Protocols in MANET Using NS2" Advances in Computer Engineering (ACE), 2010. Digital Object Identifier: 10.1109/ACE.2010.16, Publication Year: 2010, Page(s): 330 - 333.

[11] Adami, D.; Callegari, C.; Ceccarelli, D.; Giordano, S.; Pagano, M., "Design, development and validation of an NS2 module for dynamic LSP rerouting" Computer-Aided Modeling, Analysis and Design of Communication Links and Networks, 2006 11th International Workshop on Digital Object Identifier: 10.1109/CAMAD.2006.1649721, Publication Year: 2006, Page(s): 72 - 77.

[12] Qadeer, M.A.; Sharma, V.; Agarwal, A.; Husain, S.S., "Differenti- ated services with multiple random early detection algorithm using ns2 simulator" Computer Science and Information Technology, 2009. ICCSIT 2009. Digital Object Identifier: 10.1109/ICCSIT.2009.5234732, Publication Year: 2009, Page(s): 144 - 148.

[13] N. Glance, D. Snowdon and J.-L. Meunier, "Pollen: using people as a communication Medium" Computer Networks, 35(4), 2001, pp. 429 - 442.

[14] Chenna Reddy, P.; ChandraSekhar Reddy, P., "Performance Anal- ysis of Adhoc Network Routing Protocols" Ad Hoc and Ubiq- uitous Computing, 2006. ISAUHC ˝06. Digital Object Identifier: 10.1109/ISAHUC.2006.4290671, Publication Year: 2006, Page(s): 186 -187.

[15] Runcai Huang; Yiwen Zhuang; Qiying Cao, "Simulation and Analysis of MFlood Protocol in Wireless Network" Computer Science and Com- putational Technology, 2008. ISCSCT ˝08. Volume: 1, Digital Object Identifier: 10.1109/ISCSCT.2008.87, Publication Year: 2008, Page(s): 658- 662.

[16] Shijun Zhao; Panpan Wang; Junhai He, "Simulation analysis of con- gestion control in WSN based on AQM" Mechatronic Science, Elec- tric Engineering and Computer (MEC), 2011. Digital Object Identifier: 10.1109/MEC.2011.6025434, Publication Year: 2011, Page(s): 197 - 200.

[17] Jiang Hong; Yu Qing-song; Lu Hui, "Simulation and Analysis of MAC Security Based on NS2" Multimedia Information Networking and Security, 2009. MINES ˝09. Volume: 2, Digital Object Identifier: 10.1109/MINES.2009.198, Publication Year: 2009, Page(s): 502 - 505.

[18] Ruoshan Kong , "The Simulation for Network Mobility Based on NS2" Computer Science and Software Engineering, 2008 International Confer- ence on, Volume: 4, Digital Object Identifier: 10.1109/CSSE.2008.404, Publication Year: 2008, Page(s): 1070 - 1074.

