BIOINFO Security Informatics
ISSN: 2249-9423 & E-ISSN: 2249-9431, Volume 2, Issue 1, 2012, pp.-11-14.
Available online at http://www.bioinfo.in/contents.php?id=344

SECURITY AWARE ROUTING PROTOCOL FOR MANET USING ASYMMETRIC CRYPTOGRAPY USING RSA ALGORITHM

PRASAD LOKULWAR* AND VIVEK SHELKHE

Computer Science & Engineering Department, J.D. Institute of Engineering & Technology.
*Corresponding Author: Email- prasadengg16@gmail.com

Received: February 21, 2012; Accepted: March 15, 2012

Abstract- Mobile ad-hoc networks (MANTEs) are temporary networks that are built up momentarily in order to satisfy a certain emergency. Ad-hoc networks are in a great demand now a days and have a lot of advantages like emergency control, short term connections for roaming subscribers, etc. In this direction, we have designed the Ad Hoc on Demand Routing Protocol (AODV) using RSA algorithm on platform NS. Which is efficient as well as we have implemented the security technique so the we can prevent the data loss at the time of transmission. The main advantage of using the Network Simulator for the design of AODV is that we can actually observe the working of the specific protocol without the establishment of the network as NS provides the environment for the working of protocol. Which ultimately gives scope for more experimentation for applying more security techniques for getting robust network? .

Keywords- AODV protocol, Network Security, Encryption and Decryption using RSA algorithm

Citation: Prasad Lokulwar and Vivek Shelkhe (2012) Security Aware Routing Protocol for Manet using Asymmetric Cryptograpy using RSA Algorithm. BIOINFO Security Informatics, ISSN: 2249-9423 & E-ISSN: 2249-9431, Volume 2, Issue 1, pp.-11-14.

Introduction

Scope of The Thesis

WIRELESS ad hoc networks are comprised of Mobile Nodes (MNs) that are self-organizing and cooperating to ensure routing of packets among themselves. They provide robust communication in a variety of hostile environments, such as communication for the military or in disaster recovery situations when all infrastructures are down.Since the network topology of ad hoc networks is unstable and changes frequently with nodes mobility, traditional routing protocols in static networks are not efficient for ad hoc networks. Routing protocols for ad hoc networks can be classified broadly as either proactive, reactive, or hybrid (combining both behaviors).proactive protocols continuously exchange network topology information so as to constantly monitor topology changes and use that knowledge for efficient, low latency data transmission. In their turn, proactive protocols can be classified into two categories: link state routing and distance vector routing. Common proactive routing protocols include Dynamic Destination-Sequenced Distance-Vector Routing (DSDV), Optimized Link State Routing (OLSR), Multicast Optimized Link State Routing (MOLSR), etc.

Since the topologies of ad hoc networks depend on node locations and since nodes are mobiles, updates of topologies may occur more frequently than static networks, especially at high node mobility. Due to the large number of control packets required by the proactive behavior that affects the battery longevity of the MNs and restricts the goodput on the channel.

Reactive protocols were introduced to remedy the above shortcomings. These adopt a *lazy* approach to communication requirements, where nodes reacts only on-demand to data transmission requests and perform path finding operations only when needed. Reactive protocols do effectively save channel and battery power usage as they generate fewer control packets when there is no demand for transmission. The most common reactive protocols include Ad Hoc On-Demand Distance Vector Routing (AODV), Dynamic Source Routing (DSR), Source Routing-Based Multicast Protocol (SRMP), etc..

Literature Review

a. Reactive Routing Protocols

Another approach used for routing is reactive approach. This type of routing creates routes only when desired by the source node. When a node requires a route to a destination, it initiates a route discovery process within the network. This process is completed once a route is found or all possible route permutations have been examined. Once a route has been established, it is maintained by a route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired.

Example: Dynamic State Routing (DSR), Ad hoc On-Demand Distance Vector (AODV), Temporally Ordered Routing Algorithm (TORA).

b. Ad hoc On-Demand Distance Vector (AODV)

The Ad hoc On-demand Distance Vector (AODV) protocol, one of the on-demand routing algorithms that has receive the most attention, however, does not utilize multiple paths. It joins the mechanisms of DSDV and DSR. The periodic beacons, hop-by-hop routing and the sequence numbers of DSDV and the pure on-demand mechanism of Route Discovery and Route Maintenance of DSR are combined. In AODV at Every instance, route discovery is done for fresh communication which consumes more bandwidth and causes more routing over head. The source prepares RREQ packet which is broadcast to it's neighboring nodes, if neighboring node will keep backward path towards source. As soon as destination receives the RREQ packet, it sends RREP packet on received path. This RREP packet is unicast to the next node on RREP path. The intermediate node on receiving the RREP packet make reversal of path set by the RREQ packet. As soon as RREP packet is received by the source, it starts data transmission on the forward path set by RREP packet. Sometimes while data transmission is going on, if path break occurs due to mobility of node out of coverage area of nodes on the active path, data packets will be lost. When the network track requires real time delivery (voice, for instance), dropping data packets at the intermediate nodes can be costly. Likewise, if the session is a best effort, TCP connection, packet drops may lead to slow start, timeout, and throughput degradation.

Proposed Mechanism

a. Security Aware AODV Using RSA Algorithm

Introduction

Encryption is the act of encoding text so that others not privy to the decryption mechanism (the "key") cannot understand the content of the text. Encryption has long been the domain of spies and diplomats, but recently it has moved into the public eye with the concern of the protection of electronic transmissions and digitally stored data. Standard encryption methods usually have two basic flaws: A secure channel must be established at some point so that the sender may exchange the decoding key with the receiver; and There is no guarantee who sent a given message. Public key encryption has rapidly grown in popularity (and controversy, see, for example, discussions of the Clipper chip on the archives given below) because it offers a very secure encryption method that addresses these concerns. In a classic cryptosystem in order to make sure that nobody, except the intended recipient, deciphers the message, the people involved had to strive to keep the key secret. In a public-key cryptosystem. The public key cryptography solves one of the most vexing problems of all prior cryptography: the necessity of establishing a secure channel for the exchange of the key.

In cryptography, RSA (which stands for Rivest, Shamir and Adleman who first publicly described it) is an algorithm for public-key cryptography. It is the first algorithm known to be suitable for signing as well as encryption, and was one of the first great advances in public key cryptography. RSA is widely used in electronic commerce protocols, and is believed to be secure given sufficiently long keys and the use of up-to-date implementations.

Operation

The RSA algorithm involves three steps: key generation, encryption and decryption.

Key Generation

RSA involves a public key and a private key**.** The public key can be known to everyone and is used for encrypting messages. Messages encrypted with the public key can only be decrypted using the private key. The keys for the RSA algorithm are generated the following way:

Choose two distinct prime numbers *p* and *q*.For security purposes, the integers *p* and *q* should be chosen uniformly at random and should be of similar bit-length. Prime integers can be efficiently found using a primality test.

1. Compute $n = pq$.
   - *n* is used as the modulus for both the public and private keys
2. Compute $\varphi(pq) = (p - 1)(q - 1)$. ($\varphi$ is Euler's totient function).
3. Choose an integer *e* such that $1 < e < \varphi(pq)$, and *e* and $\varphi(pq)$ share no divisors other than 1 (i.e. *e* and $\varphi(pq)$ are coprime).
   - *e* is released as the public key exponent.
   - *e* having a short bit-length and small Hamming weight results in more efficient encryption. However, small values of *e* (such as $e = 3$) have been shown to be less secure in some settings.
4. Determine *d* (using modular arithmetic) which satisfies the congruence relation. $de \equiv 1 \pmod{\varphi(pq)}$
   - Stated differently, $ed - 1$ can be evenly divided by the quotient $(p - 1)(q - 1)$
   - This is often computed using the extended Euclidean algorithm.
   - *d* is kept as the private key exponent

The public key consists of the modulus *n* and the public (or encryption) exponent *e*. The private key consists of the private (or decryption) exponent *d* which must be kept secret..

Encryption

Destination node transmits its public key (*n*,*e*) to Source node and keeps the private key secret. then source wants to send message **M** to Destination

It first turns **M** into an integer $0 < m < n$ by using an agreed-upon reversible protocol known as a padding scheme. It then computes the cipher text *c* corresponding to:

$$c = m^e \bmod n \quad (1)$$

This can be done quickly using the method of exponentiation by squaring. Source then transmits *c* to Destination.

Decryption

Destination can recover *m* from *c* by using her private key exponent *d* by the following computation:

$$c^d \equiv m \pmod{n}. \tag{2}$$

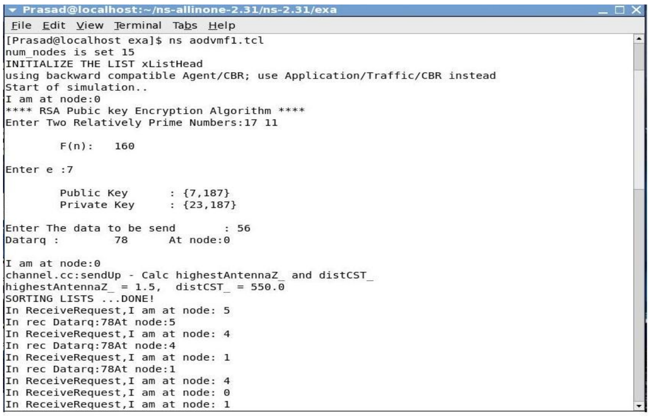
Given *m*, Destination can recover the original message **M** by reversing the padding scheme.

Example of RSA Algorithm

Example of RSA with small numbers:
p = 47, *q* = *71*, compute *n = pq = 3337*
Compute phi = 46 * 70 = 3220
Let *e* be 79, compute d = *79-1 mod 3220 = 1019*
Public key is *n* and *e,* private key *d,* discard *p* and *q.*
Encrypt message *m = 688, 68879 mod 3337 = 1570 = c.*
Decrypt message c = *1570, 15701019 mod 3337 = 688 = m.*
Thus RSA is very useful algorithm in order to obtain the security aware AODV protocol as it uses both the public key as well as the private key.

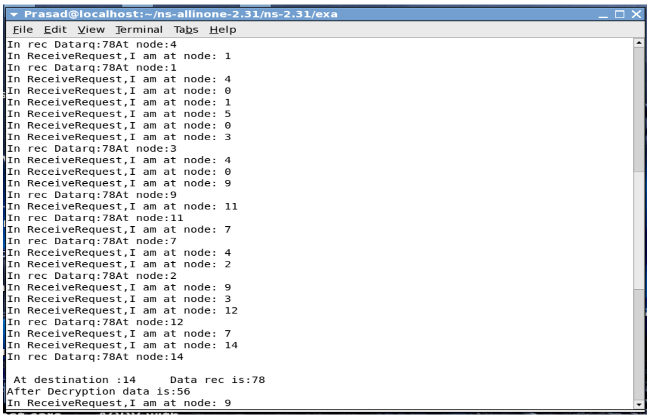
Testing

Technical investigations conducted for our project are self explained with respective snapshots


Prasad@localhost:~/ns-allinone-2.31/ns-2.31/exa
File Edit View Terminal Tabs Help

```
[Prasad@localhost exa]$ ns aodvmf1.tcl
num_nodes is set 15
INITIALIZE THE LIST xListHead
using backward compatible Agent/CBR; use Application/Traffic/CBR instead
Start of simulation..
I am at node:0
**** RSA Pubic key Encryption Algorithm ****
Enter Two Relatively Prime Numbers:17 11

        F(n):   160

Enter e :7

        Public Key      : {7,187}
        Private Key     : {23,187}

Enter The data to be send       : 56
Datarq :        78      At node:0

I am at node:0
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
In ReceiveRequest,I am at node: 5
In rec Datarq:78At node:5
In ReceiveRequest,I am at node: 4
In rec Datarq:78At node:4
In ReceiveRequest,I am at node: 1
In rec Datarq:78At node:1
In ReceiveRequest,I am at node: 4
In ReceiveRequest,I am at node: 0
In ReceiveRequest,I am at node: 1
```

Fig. 4.1- snapshot of terminal showing execution of aodv. tcl (Encrypting Data)


Prasad@localhost:~/ns-allinone-2.31/ns-2.31/exa
File Edit View Terminal Tabs Help

```
In rec Datarq:78At node:4
In ReceiveRequest,I am at node: 1
In rec Datarq:78At node:1
In ReceiveRequest,I am at node: 4
In ReceiveRequest,I am at node: 0
In ReceiveRequest,I am at node: 1
In ReceiveRequest,I am at node: 5
In ReceiveRequest,I am at node: 0
In ReceiveRequest,I am at node: 3
In rec Datarq:78At node:3
In ReceiveRequest,I am at node: 4
In ReceiveRequest,I am at node: 0
In ReceiveRequest,I am at node: 9
In rec Datarq:78At node:9
In ReceiveRequest,I am at node: 11
In rec Datarq:78At node:11
In ReceiveRequest,I am at node: 7
In rec Datarq:78At node:7
In ReceiveRequest,I am at node: 4
In ReceiveRequest,I am at node: 2
In rec Datarq:78At node:2
In ReceiveRequest,I am at node: 9
In ReceiveRequest,I am at node: 3
In ReceiveRequest,I am at node: 12
In rec Datarq:78At node:12
In ReceiveRequest,I am at node: 7
In ReceiveRequest,I am at node: 14
In rec Datarq:78At node:14

 At destination :14     Data rec is:78
After Decryption data is:56
In ReceiveRequest,I am at node: 9
```

Fig. 4.2- Snapshot of terminal showing execution of aodv.tcl (Decrypting data)

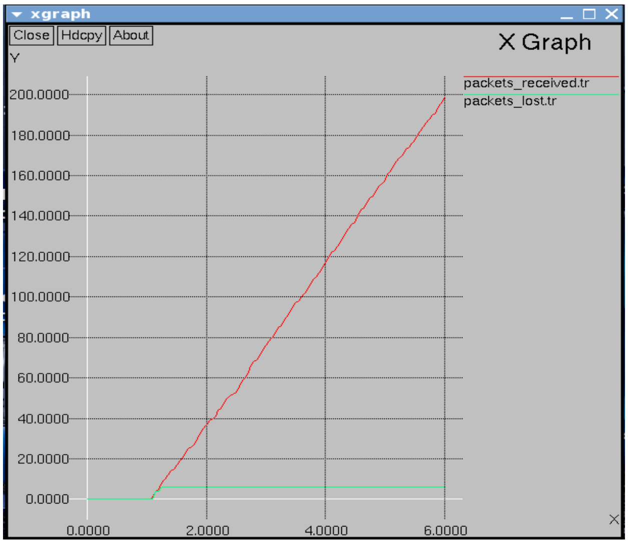


Fig. 4.3- Snapshot of terminal showing execution of xgraph of aodv

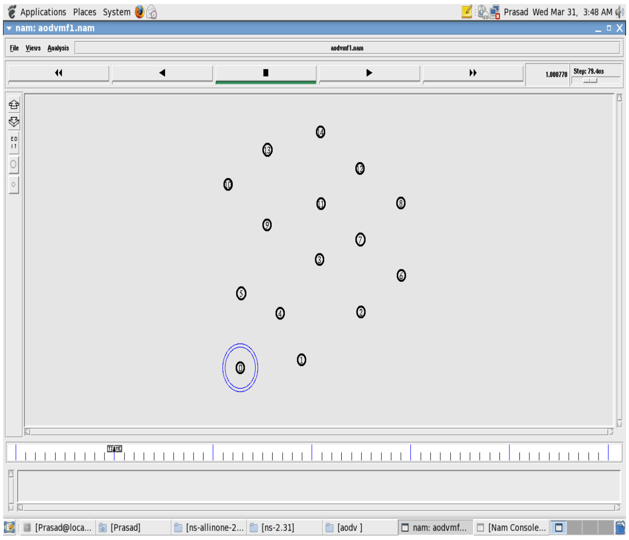


Fig. 4.4- Snapshot of nodes start the broadcasting

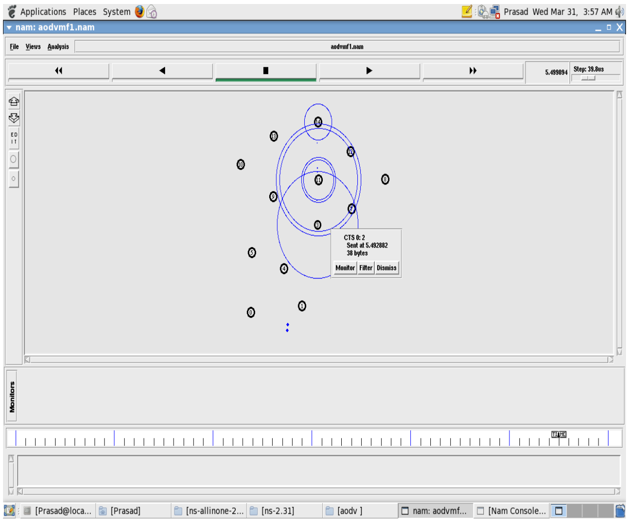


Fig. 4.5- Snapshot shows the node range

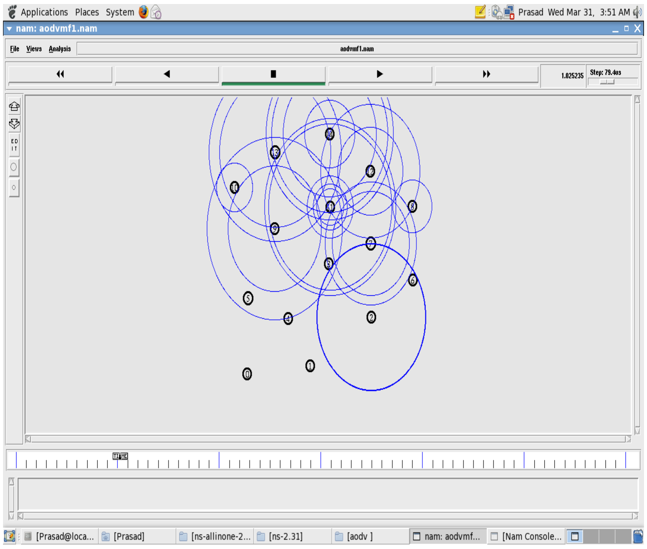

Applications Places System
Prasad Wed Mar 31, 3:51 AM
nam: aodvmf1.nam
File Views Analysis
aodvmf1.nam
1.025235
Step: 79.4ms
[Prasad@loca... [Prasad] [ns-allinone-2... [ns-2.31] [aodv] nam: aodvmf... [Nam Console...

Fig. 4.6- Snapshot of multiple node broadcasting to finding a route

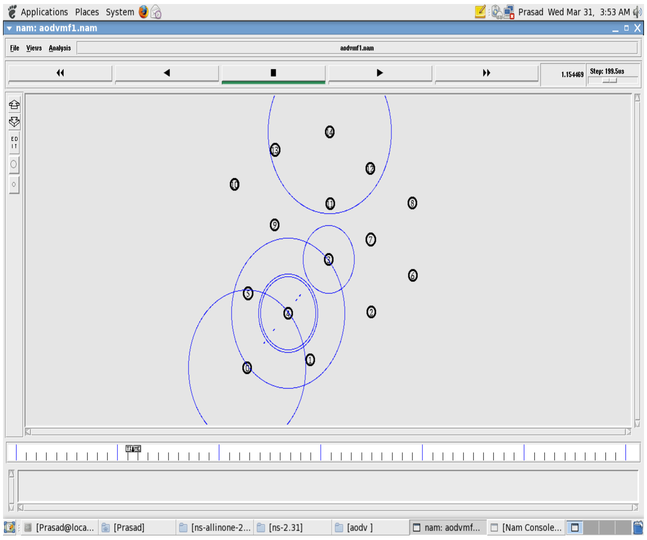
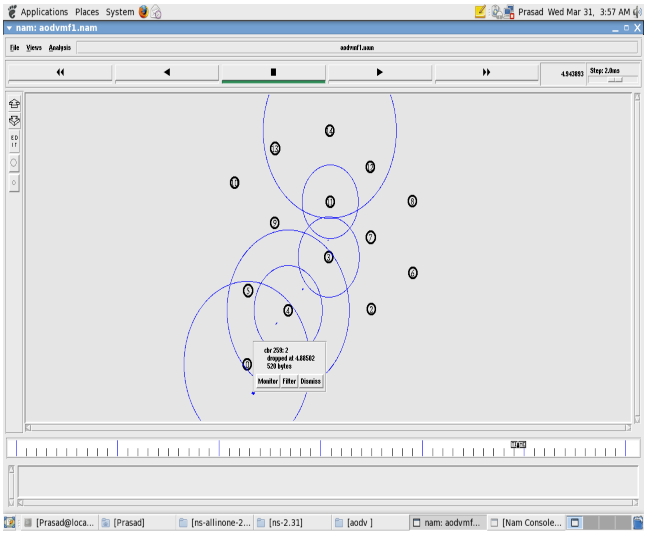

Applications Places System
Prasad Wed Mar 31, 3:53 AM
nam: aodvmf1.nam
File Views Analysis
aodvmf1.nam
1.154469
Step: 199.5us
[Prasad@loca... [Prasad] [ns-allinone-2... [ns-2.31] [aodv] nam: aodvmf... [Nam Console...

Fig. 4.7- Snapshot of aodv protocol send a packet from source to destination


Applications Places System
Prasad Wed Mar 31, 3:57 AM
nam: aodvmf1.nam
File Views Analysis
aodvmf1.nam
4.943893
Step: 2.0ms
Monitor
Filter
Dismiss
[Prasad@loca... [Prasad] [ns-allinone-2... [ns-2.31] [aodv] nam: aodvmf... [Nam Console...

```
cbr 259: 2
dropped at 4.88502
520 bytes
```

Fig. 4.8- Snapshot of drop the packet

Conclusion

In this paper, we design a security to the protocol to provide reliable efficient data transfer. Here we implement the Ad hoc On Demand Distance Vector protocol and provide the security by using Asymmetric technique. The AODV network protocol establish at the time of broadcasting. To prevent the data loss and misuse of data we have implemented the security using Asymmetric technique. The encryption and decryption are used for the security in AODV protocol. The Asymmetric technique uses the RSA encryption method for the encoding of the data to be sent. Thus with the use of broadcasting methods of AODV the network is established and data packets are sent to the destination nodes.

Acknowledgement

A project is creative work of many minds. A proper synchronization between individuals is must for any project to be completed successfully. We are extremely grateful to all the helping contributors for the successful completion of our project.

I express my sincere gratitude to *Resp. p. M. Jawandhiya* Head of the Department, Computer Science and Engineering.

References

[1] Charles E. Perkins, Elizabeth M. (2003) *RFC* 3561, *IETF*, rfc356.txt.

[2] Law and Kelton W. (2000) *McGraw-Hill*.

[3] CMU Monarch Group, CMU Monarch extensions to the NS-2 simulator, *http://monarch.cs.cmu.edu/cmu-ns.html*.

[4] Fall K., and Varadhan K (2003) *http://www.isi.edu/nsnam/ns/ns-documentation.html*.

[5] Perkins C.E. and Royer E.M. (2001) *In Ad Hoc Networking*, 173-219.

[6] Perkins C. *Ad hoc On Demand Distance Vector (AODV) Routing*, Internet draft, draft-ietfmanet- aodv-00.txt.

[7] Perkins C.E. and Royer E.M. (1999) *In Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 90-100, IEEE Computer Society.

[8] Johnson D.B. and Maltz D.A. (1996) *Ad Hoc Wireless Networks, Mobile Computing,* 5, 153-181, Kluwer Academic Publishers.