Volume 2, Issue 6, June 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Algorithm for Crypto Analysis in Manet

Er. Atul Saini[*]
D.C.S.A & KUK
atulsaini31@gmail.com

Dr. P.K Suri
Prof. and Chairman,HCTM *,Kaithal*
pksuritf25@yahoo.com

Abstract— **A mobile ad hoc network (MANET) is self-organizing, dynamic topology network, which is formed by a collection of mobile nodes through radio links. MANETs enable wireless communication between mobile devices without relying on a fixed infrastructure. Hence, routing in dynamic network is a new challenge. We used various routing algorithms for smooth exchange of information between mobile nodes. Generally we imply security on all the nodes of the network. But this causes wastage of time and cost. This paper proposes that first we have to find out the shortest path and then imply the security in multicast routing. The routing is done with the help of GDH and the Encryption and Decryption of data is done with the help of a new Cryptographic technique that i have proposed in this paper. This proposed scheme will improve the performance of the network such as delay and packet delivery ratio than traditional routing algorithms.**

Keywords— **Mobile ad hoc network, group key management, multicast, GDH, Cryptography.**

I. Introduction

Mobile ad hoc network (MANET) consists of nodes which are connected by wireless links, where each node communicates with other nodes directly or indirectly through intermediate nodes. Thus, all nodes in a MANET basically function as mobile routers participating in some routing protocol required for deciding and maintaining the routes. Routing in MANETs is challenging since there is no central coordinator that manage routing decisions. Routing is one of the key issues in MANETs due to their highly dynamic and distributed nature. Numerous ad hoc routing algorithms exist to allow networking under various conditions. They can be separated into three groups, proactive, reactive and hybrid algorithms. In proactive routing algorithms maintain continuously updated state of the network and the existing routes; however, in some cases it may generate an unnecessary overhead to maintain the routing tables and then may be better to create routes only on demand, the case of reactive routing algorithms. In reactive routing algorithms require time-consuming route creations that may delay the actual transmission of the data when sources have no path towards their destination and then, in this case may be better to use a proactive routing algorithm. In hybrid protocols try to profit the advantages of both reactive and proactive protocols and combine their basic properties into one. These protocols have the potential to provide higher scalability than pure reactive or proactive protocols thanks to the collaboration between nodes with close proximity to work together and therefore reduce the route discovery overhead Multiple routing protocols have been developed for MANETs.In proactive\ protocols, every node maintains the network topology information in the form of routing tables by periodically exchanging routing information. Routing\ information is generally flooded in the whole network. Whenever a node requires a path to a destination, it runs an appropriate path finding algorithm on the topology information it maintains. The destination sequenced distance vector routing (DSDV) protocol, and wireless routing protocol (WRP) are some examples for the proactive protocols.

Reactive protocols do not maintain the network topology information. They obtain the necessary path when it is required, by using a connection establishment process. Hence these protocols do not exchange routing information periodically. The dynamic source routing (DSR), Ad-hoc on-de source routing (DSR), Ad-hoc on-demand distance vector routing (AODV), and temporally ordered routing (TORA) algorithm are some examples for the protocols that belong to this category. In proactive routing algorithms maintain continuously updated state of the network and the existing routes; however, in some cases it may generate an unnecessary overhead to maintain the routing tables and then may be better to create routes only on demand, the case of reactive routing algorithms. In reactive routing algorithms require time-consuming route creations that may delay the actual transmission of the data when sources have no path towards their destination and then, in this case may be better to use a proactive routing algorithm. In hybrid protocols try to profit the advantages of both reactive and proactive protocols and combine their basic properties into one.

II. Distributed Bellman-Ford

The DBF algorithm was developed originally to support routing in the ARPANET. A version of it is known as Routing Internet Protocol (RIP)[1] and is still being used today to support routing in some Internet domains. It is a table-driven routing protoco1, that is, each router constantly maintains an up-to-date routing table with information on how to reach all possible destinations in the network. For each entry the next router to reach the destination and a metric to the destination are recorded. The metric can be hop distance, total delay, or cost of sending the message. Each node in the network begins by informing its neighbors about its distance to all other nodes. The receiving nodes extract this information and modify their routing table if any route measure has changed. For instance, a different route may have been chosen as the best route or the metric to the destination may have been altered. The node uses the following formula to calculate the best route:

D(i, j)* = min [d(i, *k)* + *D*(*k*, *j*) *]

where *D(i,j)* is the metric on the "shortest" path from node ***i*** to node *j* , *d*(*i*, *k)* is the cost of traversing directly from node ***i*** to node ***k*,** and *k* is one of the neighbors of node ***i*.** After recomputing the metrics, nodes pass their own distance information to their neighbor nodes again. After a while, all nodes/routers in the network have a consistent routing table to all other nodes.

This protocol does not scale well to large networks due to a number of reasons. One is the so-called count-to-infinity problem. In unfavorable circumstances, it takes up to *N* iterations to detect the fact that a node is disconnected, where *N* is the number of nodes in the network [2]. Another problem is the increase of route update overhead with mobility. RIP uses time triggered (periodic, about a 30-s interval) and event-triggered (link changes or router failures) routing updates. Mobility can be expressed as rate of link changes and/or router failures. In a mobile network environment, event-triggered routing updates tend to outnumber time-triggered ones, leading to excessive overhead and inefficient usage of the limited wireless bandwidth.

III. Diffie-Hellman Two-Party Agreement(DH)

This basic protocol, proposed in a landmark paper [3], allows two nodes to build a common key. The principal of this protocol is simple: the two involved nodes, M_1 and M_2, send one another a partial key to be used for the common key computation. M_1 generates a random number $r1$ ($1 \leq r1 \leq p$), and sends α^{r1} to M_2, such that α and *p* are constants known by each node. On the other hand, M_2 generates a random number r_2, and sends α^{r2} to *M*1. Thereby, each node could compute the common key, which is $\alpha^{r1\times r2}$ This solution is based on discrete logarithmic arithmetic, and also relies on the agreement on the parameters α and *p* between the two nodes. Although it is simple and limited to two nodes' common key establishment, this protocol was used to design more sophisticated protocols, as we will see later.

Example .[5] This protocol uses exponentiation to share a secret between two parties, Alice and Bob. The protocol involves an initiator, Alice, and a responder, Bob. We use the common notation A → B : M to stand for "A sends message M to B". Raising message M to the power of exponent X is denoted by $(M)^X$. There is a public term denoted by g, which will be the base of our exponentiations. We represent the product of exponents by using the symbol *. Nonces are represented by N_X, denoting a nonce created by principal X. The protocol description is as follows.

1. A → B : A
   Alice sends her name to Bob.
2. A → B : B
   Alice sends Bob's name to Bob.
3. A → B : g^{N_A}
   Alice creates a new nonce N_A and sends g^{N_A} to Bob.
4. B → A : B
   Bob sends his name to Alice.
5. B → A : A
   Bob sends Alice's name to herself.
6. B → A : g^{N_B}
   Bob creates a new nonce N_B and sends g^{N_B} to Alice.

Intuitively, when Bob receives g^{N_A}, he raises it to the N_B, to obtain $g^{N_A N_B} = g^{N_A * N_B}$
Likewise, when Alice receives g^{N_B}, she raises it to the N_A, to obtain $g^{N_B N_A} = g^{N_B * N_A}$.
And due to the commutativity of the symbol *, they know the equivalence $g^{N_B * N_A} = g^{N_A * N_B}$. An observer of the exchange who does not know N_A nor N_B cannot find $g^{N_A * N_B}$, and so Alice and Bob have computed a shared secret, i.e., $g^{N_B * N_A}$. Of course, the attacker can always learn a term $g^{(N_A * N_I)}$, where N_I is a nonce created by the intruder, even by using a passive intruder model. The point is that he can also make believe to Alice that $g^{(N_A * N_I)}$ is the shared key she is sharing with Bob. This is usually modelled by adding to the protocol a new message where Alice sends to Bob some secret, encrypted by $g^{(N_A * N_I)}$. Existence of an attack is expressed by saying that the attacker can obtain this secret. For the sake of simplicity and because we are focused in AC-theories, we omit this last part of the protocol and concentrate just in whether the intruder can learn X^{N_A} for some exponentiation X, where X^{N_A} is the key calculated by Alice.
In a rule-based representation of this protocol, parts of a received message whose make-up cannot be verified by a principal are represented by variables. That is, since nonces are known only to the principal who generated it, and retrieving the nonce would require the computation of a discrete logarithm, we say that Bob receives a variable X of a generic message sort instead of g^{N_A} and similarly for Alice. The symbol * is associative and commutative and satisfies the following additional property with respect to exponentiation:
$(X^Y)^Z = X^{(Y*Z)}$
The intruder abilities to create, manipulate, and delete messages according to the Dolev-Yao attackers capabilities [6] are described as follows, where we use the special symbol _ƐT̲ to represent that the intruder knows something, and I denotes the intruder's name:

$$\frac{M_1 \in \underline{T}, M_2 \in \underline{T}}{(M_1 * M_2) \in \underline{T}} \qquad \frac{X \in \underline{T}, Y \in \underline{T}}{X^Y \in \underline{T}} \qquad \frac{}{N_1 \in \underline{T}}$$

The intruder also knows the names of all the principals and the base g.

If we ask ourselves whether the intruder can learn a message X^{N_A} for some variable X received by Alice (representing the nonce that Alice receives from Bob), the answer is yes for an infinite set of instances for X, e.g., g^{N_I}, $(g^{N_I})^{N_I'}$, $((g^{N_I})^{N_I'})^{N_I''}$, etc. If we take instantiation X→ g^{N_I} , the intruder can learn the message $g^{(N_A * N_I)}$ by means of the following sequence of actions (only the three first steps are necessary but we need Alice to complete the protocol in order to believe she is sharing a shared key with Bob:

1. A→B : A
   Alice sends her name to Bob, but it is intercepted by the intruder.
2. A→B : B
   Alice sends Bob's name to Bob, but it is intercepted by the intruder.
3. A →B : g^{N_A}
   Alice creates a new nonce N_A and sends g^{N_A} to Bob, but it is intercepted by the intruder.
4. I →A : B
   The intruder sends Bob's name to Alice.
5. I→ A : A
   The intruder sends Alice's name to Alice.
6. I → A : g^{N_I}
   The intruder creates a new nonce N_I and sends g^{N_I} to Alice.

The intruder is able to learn the message $g^{(N_A * N_I)}$ just by raising the intercepted message g^{N_A} to N_I . Note that the intruder does not need to know N_A, since he gets the desired effect thanks to the equational properties for exponentiation and product of exponents described above.

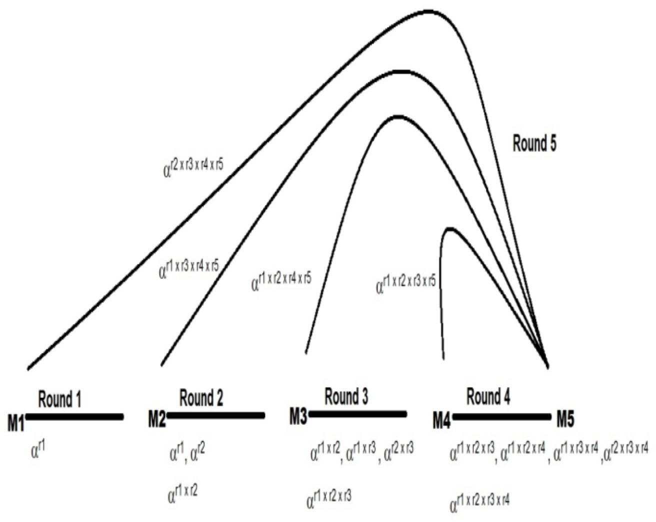
IV. GENERAL DIFFIE-HELLMAN (GDH)

Steiner *et al.* [4] proposed a n-party generalization of the basic two-party DH protocol (described before). The new protocol consists of n rounds, allowing n nodes to establish a common key. In the first $n - 1$ rounds contributions are collected from each node. In the first round, M_1 generates $r1$ and computes α^{r1}, which it sends to M_2. In the second step M_2 generates r_2, computes α^{r2} and sends it to M_3, along with α^{r1} and $\alpha^{r1\times r2}$. This latter sends to $M4$ (after making the required computations) the third-round partial factors, i.e., $\alpha^{r1\times r2}$, $\alpha^{r1\times r3}$, $\alpha^{r2\times r3}$, as well as the third-round partial key $\alpha^{r1\times r2\times r3}$. This process continues for each M_i ($i < n$). Upon the $(n - 1)$th round, the collector node M_n receives the $(n - 1)$th round partial factors, and the $(n - 1)$th round partial key, then it generates its random number and computes the final key K. In the last round, node M_n sends each M_i the appropriate (n)th round partial factor, i.e.

$$K = \alpha^{(\pi_{j=1}^{n} r_j)/r_j}$$

Consequently, each node uses its random number to compute the common key K. Note that partial factors are used to avoid sending the final resulted key during the last round. Also note that the $(n - 1)$th round requires $n - 1$ operations (sending the partial factor to each node), which makes the computational complexity of the solution 0 (2 × (n – 1)).

Even though it uses a collector, this solution is contributory, since each node contributes to the key computation with the random number it generates. Nevertheless, the major drawback of this solution is the important overhead, due to the message size rising from round to round. This can also cause a problem with scalability.


Round 5
$\alpha^{r2\times r3\times r4\times r5}$
$\alpha^{r1\times r3\times r4\times r5}$
$\alpha^{r1\times r2\times r4\times r5}$
$\alpha^{r1\times r2\times r3\times r5}$
Round 1
Round 2
Round 3
Round 4
M1
M2
M3
M4
M5
α^{r1}
α^{r1}, α^{r2}
$\alpha^{r1\times r2}$
$\alpha^{r1\times r2}, \alpha^{r1\times r3}, \alpha^{r2\times r3}$
$\alpha^{r1\times r2\times r3}$
$\alpha^{r1\times r2\times r3}, \alpha^{r1\times r2\times r4}, \alpha^{r1\times r3\times r4}, \alpha^{r2\times r3\times r4}$
$\alpha^{r1\times r2\times r3\times r4}$

Example of General Diffie-Hellman with five Node

V. PROPOSED WORK:

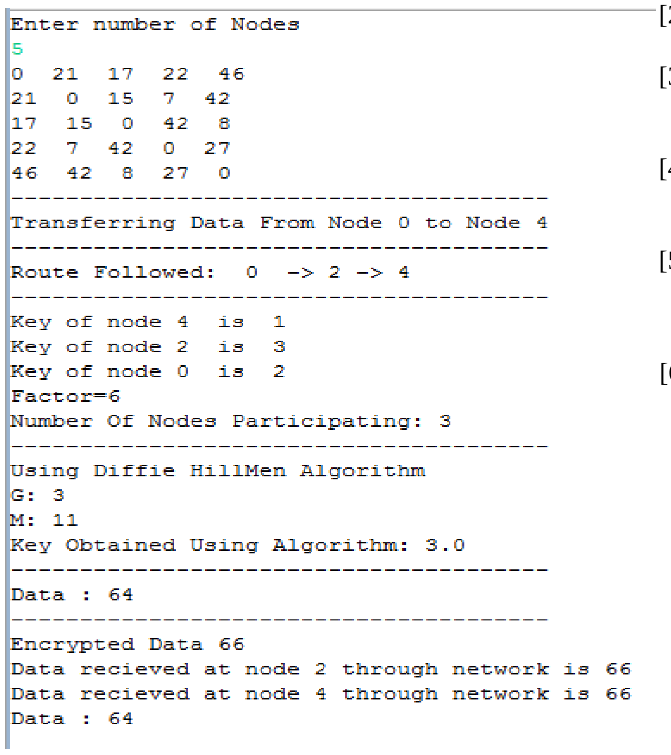
Generally we imply security on all the nodes of the network. But this causes wastage of time and cost. This paper proposes that first we have to find out the shortest path and then imply the security in multicast routing. The routing is done with the help of GDH and the Encryption and Decryption of data is done with the help of a new Cryptographic technique that i have proposed in this paper. This proposed scheme will improve the performance of the network such as delay and packet delivery ratio than traditional routing algorithms:

- A network is randomly created
- Then the shortest distance from source to destination is found with the help of bellmen ford algorithm
- Apply GDH on each node starting from source to destination node on the shortest path calculated.
- Apply Encryption and decryption scheme over the DATA from source to destination node.

Encryption:	*Decryption:*
DATA + N = A;	D – N = X;
A * N = B;	X * N = Y;
B - N = C;	Y + N = Z;
C / N = D;	C / N = DATA;

where N=GDH key;
D=encrypted data

VI. Experiment Results And Analysis:



```
Enter number of Nodes
5
0  21  17  22  46
21  0  15  7  42
17  15  0  42  8
22  7  42  0  27
46  42  8  27  0
---------------------------------------
Transferring Data From Node 0 to Node 4
---------------------------------------
Route Followed:  0  -> 2 -> 4
---------------------------------------
Key of node 4  is  1
Key of node 2  is  3
Key of node 0  is  2
Factor=6
Number Of Nodes Participating: 3
---------------------------------------
Using Diffie HillMen Algorithm
G: 3
M: 11
Key Obtained Using Algorithm: 3.0
---------------------------------------
Data : 64
---------------------------------------
Encrypted Data 66
Data recieved at node 2 through network is 66
Data recieved at node 4 through network is 66
Data : 64
```

In the last we can see that by applying this algorithm first of all we have found the shortest path between the source and destination node in a generalized network and then applied the security over it with the help of GDH and then finally did the encryption and decryption of the whole data that is to be sent over the network.

VII. Conclusion:

Case I: If there is no intermediate node in between the source and the destination node. Then their is no way that the key shared between the two nodes gets intended and the data transfer is secure.
Case II: if there are some intermediate node in between the source and the destination node i.e along the path between the source and the destination node and if some maliaous node and if some maliaous node intrudes the key and change it and then forwards it to the destined node waiting for the key to be shared, the destined node will not be able to know whether the key coming to be shared is either intruded or not and take it as a key that is not infected from any malicious node(intruder) .

VIII. Future Work:-

If Case II occurs then the prime target is to detect the malicious node along the whole path and to make the path secure so that data can be transferred securely over the path without any intrusion.

IX. REFERENCES

[1] G. Malkin, "RIP Version 2 - Carrying Additional information," Internet Draft, drah-ietf-ripv2-protocol-v2-05.txt, Jun. 1998, work in progress
[2] A.*S.* Tanenbaum, Computer Networks, 3rd ed., Upper Saddle River, NJ: Prentice Hall. Mar. 1996.
[3] W. Diffie and M. E. Hellman, “New Directions in Cryptography,”*IEEE Trans. Info Theory*, vol. IT-22, no. 6, 1976, pp. 644–54
[4] M. Steiner, G. Tsudik, and M. Waidner, “Diffie Hellman Key Distribution Extended to Group Communication,” *ACM Conf. Comp. and Commun. Security*, 1996
[5] Santiago Escobar, Joe Hendrix, Catherine Meadows, and Jos´e Meseguer, “Diffie-Hellman Cryptographic Reasoning in the Maude-NRL Protocol Analyzer”

[6] D. Dolev and A. Yao. On the security of public key protocols. IEEE Transaction on Information Theory, 29(2):198–208, 1983.