ISSN:0975-9646

# Encoding N-party Man-In-Middle Attack for Diffie–Hellman Algorithm in a Client-Server Paradigm

Sulochana Devi[# 1] Ritu Makani[*2]

*#1Student of Masters of Technology, Department of Computer Science and Engineering*
*Guru Jambeshwar University of Science & Technology Hisar, India*

**2Assistant Professor, Department of Computer Science and Engineering*
*Guru Jambeshwar University of Science & Technology Hisar, India*

*Abstract* — **Diffie–Hellman key exchange (D–H) is a specific method of exchanging cryptographic keys. It is one of the earliest practical methods of key exchange implemented within the field of cryptography. In the original description, the Diffie–Hellman exchange by itself does not provide authentication of the communicating parties and is thus vulnerable to a man-in-the-middle attack. There may be many attackers between sender and receiver. This paper mainly focuses on generation of N-Party Man-in-Middle Attack in Diffie–Hellman Key Exchange Protocol. We have analyzed how this method can work in Diffie-Hellman Key-Exchange Protocol for sharing secret key between users when more than one attacker are present between sender and receiver.**

***Keywords*— Diffie-Hellman Algorithm, encryption, decryption, asymmetric, cryptography.**

## I. Introduction

For information to be secure, one has to be prevent access to it from unauthorized users, prevent it from undergoing unwanted changes, and ensure that it is available to its intended users. This can be guaranteed by means of protocols that make use of security primitives such as encryption, digital signatures and hashing. Cryptography and encryption/decryption methods fall into two broad categories: symmetric and public key. In symmetric cryptography, parties share same encryption/decryption key. Therefore, before using a symmetric cryptography system, the users must somehow come to an agreement on a key to use. An obvious problem arises when the parties are separated by large distances, which is commonplace in today's worldwide digital communications. If the parties did not meet prior to their separation, how do they agree on the common key to use in their cryptosystem without a secure channel? They could send a trusted courier to exchange keys, but that is not feasible, if time is a critical factor in their communication. The first researchers to formulate and publish the concepts of public-key cryptography were Whitfield Diffie and Martin Hellman, both from Stanford University, in parallel with Ralph Merkle, from the University of California at Berkeley. Specifically, Diffie and Hellman worked on public key cryptography while Merkle made his contributions on public key distribution [6]. When they became aware of each other's work they decided to work together in hope for better results. This later led to the publication of their joint paper, titled "New Directions in Cryptography"—published in November 1976. This paper brought a new idea to the field of cryptography; it described the key concepts of public-key cryptography that is called Diffie-Hellman key exchange protocol [15]. This algorithm has a major weakness in the form of man-in-the-middle attack [5]. There may be many attackers between sender and receiver, so in our proposed work we generate N-Party Man-in-Middle Attack in Diffie – Hellman Key Exchange Protocol.

## II. Literature Review

Literature review presents a number of approaches related with Diffie -Hellman Key-Exchange Protocol and provides the background to the research by describing what has been done in prior research.

Lein Harn [1] et al . This paper proposed three protocols that securely integrate Diffie–Hellman key exchange into the DSA. One-round protocol can be used in secure e- mail transmission. Two-round protocol provides authenticated key exchange for interactive communications. Three-round protocol provides authenticated, key confirmation and non playback key exchange for interactive communications.

Nan Li [4] et al. In this paper, the computational efficiency of various authentication methods are compared. Finally an improved key exchange schema based on hash function is given, which improves the security and practicality of Diffie-Hellman protocol.

Barun Biswas [6] et al. In this paper a new technique is introduced. In the proposed technique both sender and receiver use a secret number e as the base of the log. If in the middle the key is attacked and the key is changed not necessarily the base will be e. However we can't say that man in the middle attack can be fully eliminate because the base selected by the middle man can be same as e unfortunately.

C. Krishna Kumar [7] et al. This paper states that if the key exchange takes place in certain mathematical environments, the exchange becomes vulnerable to a specific man in the middle attack. This paper explores this man in the middle attack, analyze countermeasures against the attack. The easiest method is to force authentication prior to the key exchange. Sender double encrypts a message first with own private key and then with receiver's public key.

Shilpi Gupta [8] et al. In this paper main focus on asymmetric cryptography and proposed a novel method by combining the two most popular algorithms RSA and Diffie-Hellman in order to achieve more security. RSA algorithm is used as Public key cryptography method. DH is a method for securely exchanging a shared secret between two parties, in real-time, over an un trusted network. In this approach the Diffie- Hellman is not used only for key generation but also for the generation cipher text.

Ekta Lamba [14] et al. In this paper, it is tried to give focus on the hardness of key by using safe primes that makes it almost infeasible to calculate discrete logarithms & thus using that key for encryption and decryption of data so that we get better security.

Rohini [16] et al. This paper provides harder encryption with extend public key encryption protocol for security. Proposed work in this paper provides better security and implemented in any network. It enhanced the hardness of security by DH algorithm. The DH algorithm is improved by adding modulus operation on the private key. That increases the entropy and decrease the autocorrelation.

## III. Existing System

A malicious third party (eavesdropper) retrieves sender's public component and sends his own public component to receiver. When receiver transmits his public key, third party interrupts and substitutes the value with his own public key and then sends it to sender. Now there is an agreement on a common secret key with third party instead of receiver. It is possible for third party (Man-in-Middle) to decrypt any messages sent out by sender or receiver

### A. *Algorithm for Man-in-Middle Attack*

```
i     Ave prepares for the attack by generating two random private keys $X_{M1}$ and $X_{M2}$ &
      computes the public values $Y_{M1}$ and $Y_{M2}$.
        $Y_{M1}= (r)^{XM1}$ MOD p.
        $Y_{M2}=(r)^{XM2}$ MOD p.
ii    Alice transmits $Y_A$ to Bob.
iii   Ave intercepts $Y_A$ and transmits $Y_{M1}$ to Bob.
iv    Bob transmits $Y_B$ to Alice.
v     Ave again intercepts $Y_B$ and transmits $Y_{M2}$ to Alice.
vi    Bob receives $Y_{M1}$.
vii   Alice receives $Y_{M2.}$
viii  Alice computes K1= $(Y_{M2})^{XA}$ MOD p.
ix    Bob computes key K2= $(Y_{M1})^{XB}$ MOD p.
x     Ave computes key K1= $(Y_A)^{YM2}$ MOD p.
                       K2= $(Y_B)^{YM1}$ MOD p.
```

Fig. 1 Man-in-Middle Attack

### B. *Method for Man-in-Middle Attack*

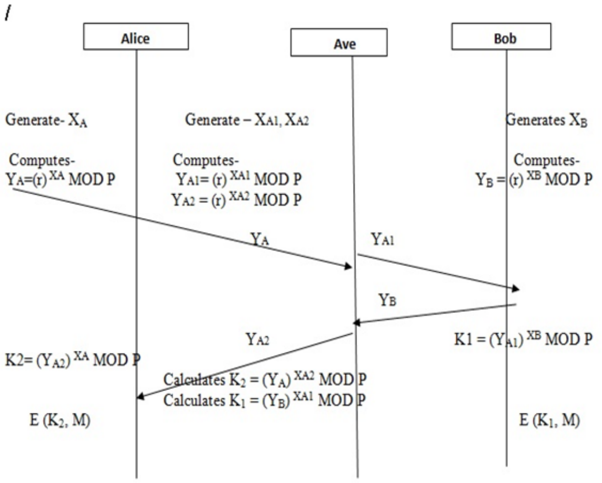

Alice
Ave
Bob
Generate- $X_A$
Generate – $X_{A1}$, $X_{A2}$
Generates $X_B$
Computes-
$Y_A$=(r) $^{XA}$ MOD P
Computes-
$Y_{A1}$= (r) $^{XA1}$ MOD P
$Y_{A2}$ = (r) $^{XA2}$ MOD P
Computes-
$Y_B$ = (r) $^{XB}$ MOD P
$Y_A$
$Y_{A1}$
$Y_B$
$Y_{A2}$
K1 = $(Y_{A1})^{XB}$ MOD P
K2= $(Y_{A2})^{XA}$ MOD P
Calculates $K_2 = (Y_A)^{XA2}$ MOD P
Calculates $K_1 = (Y_B)^{XA1}$ MOD P
E ($K_2$, M)
E ($K_1$, M)

Fig. 2 Man-in-Middle Attack

## IV. Proposed System

It is not necessary there is always one middle man; there may be many attackers between sender and receiver. When client sends his public component to server, first attacker intercept it and sends own generated first public components to server and second to client but if there is another attacker then that first component intercepted by second attacker. In this type all attackers intercept public components of their neighbor users and generate their own keys which are similar to neighbor users and decrypt the messages. Alice wants to communicate with Bob, so to generate a secret key between them; he sends his own public component to Bob. But there are many attackers between Alice and Bob. This whole process is performed in following way:-

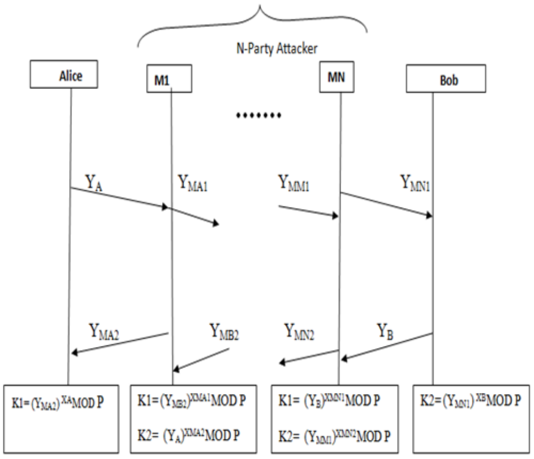


Fig. 3 N party Man-in-Middle Attack generation

## V. Implementation Of Proposed Scheme And Result

We have completed the research in following steps to get the set objectives.

1. Deep study of Diffie-Hellman Key Exchange Protocol.
2. Study of Man-in-the-Middle Attack Algorithm.

3. Proposed the algorithm for N-party Man-in-the-Middle Attack.
4. After shaping up of proposed algorithm, implement the algorithm in Ubuntu 14.04 using PERL technology.

### A. *Proposed Algorithms*

#### 1) *Algorithm at client end*

```
1   Create a new socket if :-
        i    Assign peer host.
        ii   Assign peer port number.
        iii  Assign protocol Or Error message
2   Alice computes its own public component and sends it to middle1.
        i    Take a prime number p and compute its primitive root r.
        ii   Alice take a private random number and computes public component.
        iii  Send->$socket(value)
3   Read the message sent by middle1 by using recv function.
    $socket->recv(value).
4   Compute key by using received value.
```

Middle1 works not only as client but also as server. It receives files from Alice as a receiver and sends to next user as a sender. So first we run server file and then client file at middle1 end.

#### 2) *Algorithm at Middle1 end*

```
1   Create a new socket if :-
        i    Assign peer host.
        ii   Assign peer port number.
        iii  Assign protocol or Error message
2   Wait for client connection.
3   When available, accept a new connection.
4   Middle 1 receives the data from Alice user.
5   Compute key.
6   Run client file at middle1 end to communicate with next user.
        i    Create a socket and connect to server.
        ii   Send data to next user.
        iii  Receive public component sent by middle2 user.
        iv   Compute key
```

Middle2 works not only as client but also as server. It receives files from Middle1 as a receiver and sends to next user as a sender. So first we run server file and then client file at middle2 end.

#### 3) *Algorithm at Middle2 end*

```
1   Create a new socket if :-
        i    Assign peer host.
        ii   Assign peer port number.
        iii  Assign protocol Or Error message
2   Wait for client connection.
3   When available, accept a new connection.
4   Middle2 receives the data from Middle1 user.
5   Compute key.
6   Run client file at middle2 end to communicate with next user.
7   Create a socket and connect to server.
8   Send data to next user.
9   Receive public component sent by Bob user.
10  Compute key
```

Bob works as a server. It receives files from Middle2 as a receiver and sends it to middle2. So we run server file at middle2 end.

#### 4) *Algorithm at server end*

```
1   Create a new socket if:-
        i    Assign peer host.
        ii   Assign peer port number.
        iii  Assign protocol Or Error message
2   Wait for client connection.
3   When available, accept a new connection.
4   Bob send own pubic key top middle2
5   Bob receives the data from Middle2 user.
6   Compute key.
```

### B. *Implementation and Result (Fig 4)*

Alice key generation**: -** All random numbers less than 7 can be used as private value because we consider prime number 7 to compute keys and we selected 3 as a primitive root of 7. Alice generates random private values. When private value is 3 then public component is computed as 6 and key is also 6.

Table 1 Alice key generation

| Private random values | Public components | Keys |
|---|---|---|
| 3 | 6 | 6 |
| 4 | 4 | 1 |
| 6 | 1 | 1 |
| 5 | 5 | 2 |
| 2 | 2 | 4 |
| 5 | 5 | 3 |

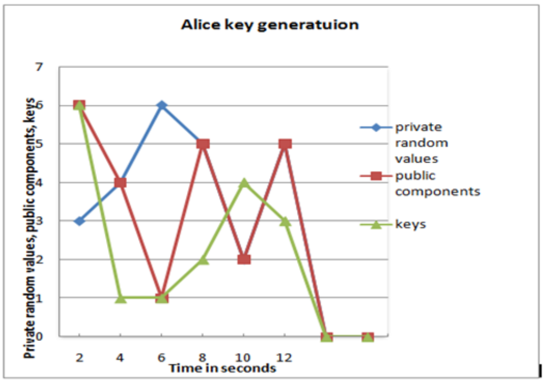
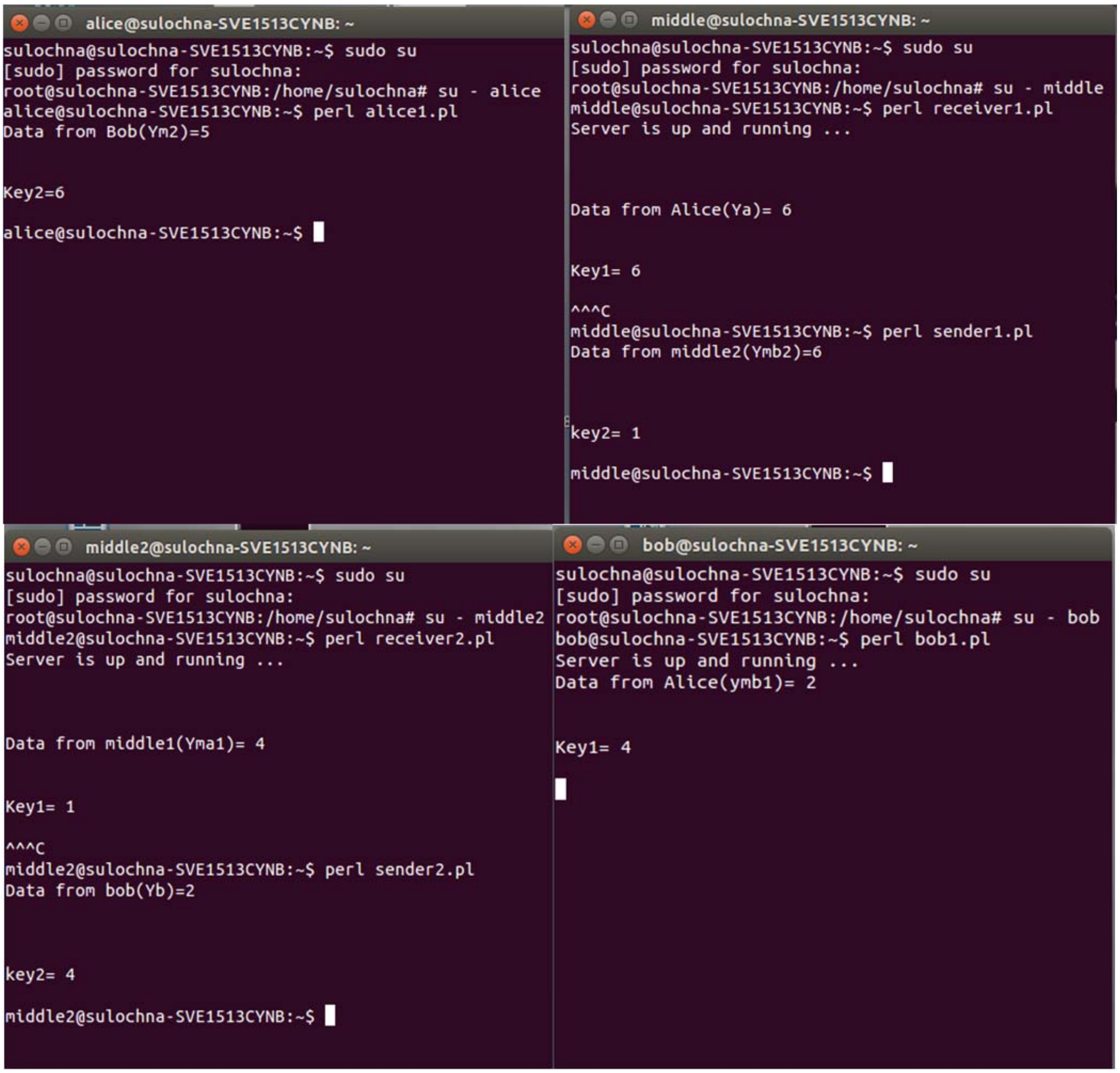


Fig. 5 Alice key generation

At specific time Alice generates random private values and computes public components. According to these values, keys are generated.


alice@sulochna-SVE1513CYNB: ~
middle@sulochna-SVE1513CYNB: ~
middle2@sulochna-SVE1513CYNB: ~
bob@sulochna-SVE1513CYNB: ~

```
sulochna@sulochna-SVE1513CYNB:~$ sudo su
[sudo] password for sulochna:
root@sulochna-SVE1513CYNB:/home/sulochna# su - alice
alice@sulochna-SVE1513CYNB:~$ perl alice1.pl
Data from Bob(Ym2)=5


Key2=6

alice@sulochna-SVE1513CYNB:~$
```

```
sulochna@sulochna-SVE1513CYNB:~$ sudo su
[sudo] password for sulochna:
root@sulochna-SVE1513CYNB:/home/sulochna# su - middle
middle@sulochna-SVE1513CYNB:~$ perl receiver1.pl
Server is up and running ...



Data from Alice(Ya)= 6


Key1= 6

^^^C
middle@sulochna-SVE1513CYNB:~$ perl sender1.pl
Data from middle2(Ymb2)=6



key2= 1

middle@sulochna-SVE1513CYNB:~$
```

```
sulochna@sulochna-SVE1513CYNB:~$ sudo su
[sudo] password for sulochna:
root@sulochna-SVE1513CYNB:/home/sulochna# su - middle2
middle2@sulochna-SVE1513CYNB:~$ perl receiver2.pl
Server is up and running ...



Data from middle1(Yma1)= 4


Key1= 1

^^^C
middle2@sulochna-SVE1513CYNB:~$ perl sender2.pl
Data from bob(Yb)=2



key2= 4

middle2@sulochna-SVE1513CYNB:~$
```

```
sulochna@sulochna-SVE1513CYNB:~$ sudo su
[sudo] password for sulochna:
root@sulochna-SVE1513CYNB:/home/sulochna# su - bob
bob@sulochna-SVE1513CYNB:~$ perl bob1.pl
Server is up and running ...
Data from Alice(ymb1)= 2


Key1= 4
```

Fig. 4  Output screen of proposed algorithm

Middle1 key generation:-

Table 2
Middle1 key generation

| Private random values | Public components | Keys |
|---|---|---|
| 4 | 4 | 6 |
| 5 | 5 | 1 |
| 5 | 5 | 1 |
| 6 | 1 | 2 |
| 2 | 2 | 1 |
| 3 | 6 | 4 |
| 3 | 6 | 2 |
| 4 | 4 | 6 |
| 1 | 3 | 4 |
| 2 | 2 | 4 |
| 2 | 2 | 3 |
| 5 | 5 | 4 |

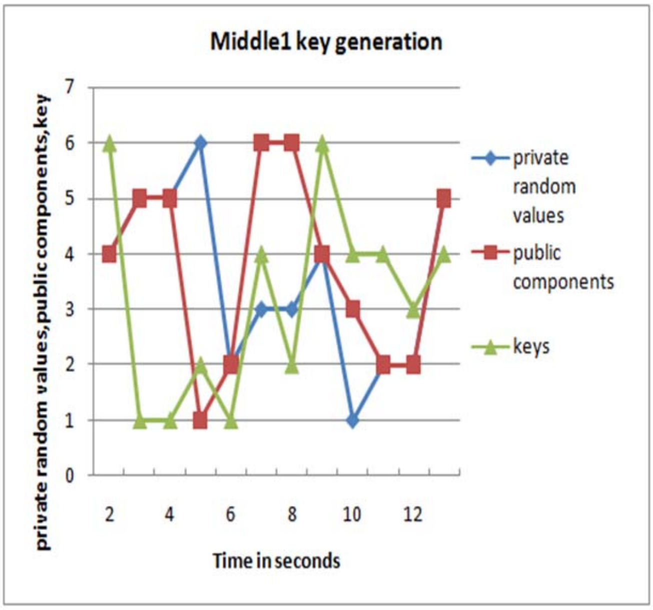


Fig. 6 Middle1 key generation

At X axis, time is shown in seconds and at a specific time Middle1 generate random private values and compute public components. According to these values, keys are generated.

Middle2 key generation

Table 3
Middle2 key generation

| Private random values | Public components | Keys |
|---|---|---|
| 2 | 2 | 1 |
| 3 | 6 | 4 |
| 3 | 6 | 2 |
| 4 | 4 | 6 |
| 1 | 3 | 4 |
| 2 | 2 | 5 |
| 2 | 2 | 6 |
| 1 | 3 | 1 |
| 5 | 5 | 4 |
| 4 | 4 | 6 |
| 4 | 4 | 4 |
| 5 | 5 | 4 |

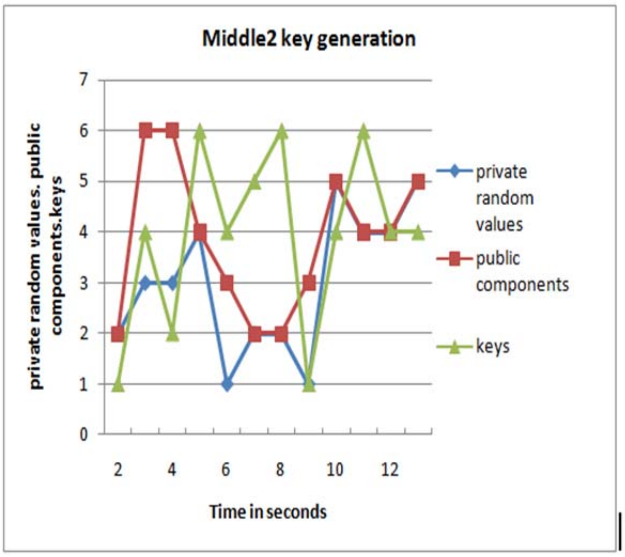


Fig. 7 Middle2 key generation

At X axis, time is shown in seconds and at a specific time Middle2 generate random private values and compute public components. According to these values keys are generated.
Bob key generation

Table 4
Bob key generation

| Private random values | Public components | Keys |
|---|---|---|
| 2 | 2 | 4 |
| 3 | 6 | 6 |
| 5 | 5 | 5 |
| 6 | 1 | 1 |
| 3 | 6 | 6 |
| 4 | 4 | 4 |

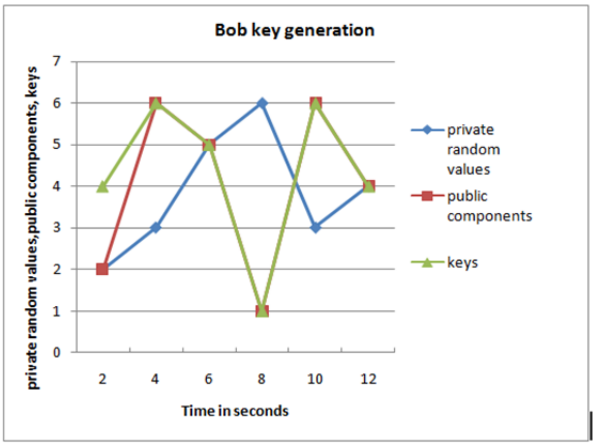


Fig. 8 Bob key generation

This type Bob also generates random private values and computes public components. According to these values keys are generated.

## VI. Conclusion And Future Scope

The Diffie-Hellman key exchange protocol is very effective scheme to generate a common secret key for both the sender and the receiver. It is used mainly wherever we need to compute key for exchange the shared key. It is not necessary there is always one middle man; there may be many attackers between sender and receiver. When client sends his public component to server, first attacker intercept it and sends own generated first public components to server and second to client but if there is another attacker then that first component intercepted by second attacker. In this type all attackers intercept public components of their neighbor users and generate their own keys which are similar to neighbor users and decrypt the messages. In our work, we generate N- party Man-in-the-Middle Attack in Diffie-Hellman key exchange protocol using PERL technology.

In the present work we considered 2-party man-in-the-middle attack in Diffie-Hellman key exchange algorithm which can be extended to N-party to create a network of intrusion. The main challenge will be to create a defense line for N-party attack and have a distributed detection system for such a scenario.

## References

[1]. Lein Harn, Manish Mehta and Wen-Jung Hsin, "*Integrating Diffie–Hellman Key Exchange into the Digital Signature Algorithm (DSA)*", IEEE communications letters, vol. 8, no. 3, March 2004.

[2]. Raphael C.-W. Phan, Member, "*Fixing the Integrated Diffie-Hellman-DSA Key Exchange Protocol* ", IEEE communications letters, vol. 9, no. 6, June 2005.

[3]. Ik Rae Jeong, Jeong Ok Kwon, and Dong Hoon Lee, "*Strong Diffie-Hellman-DSA Key Exchange*", IEEE communications letters, vol. 11, no. 5, may 2007.

[4]. Nan Li, "*Research on Diffie-Hellman Key Exchange Protocol*", 2nd International Conference on Computer Engineering and Technology, Vol 5, 2010.

[5]. Barun Biswas, Krishnendu Basuli, " *A novel process for key exchange avoiding man-in-middle attack*", International Journal of Advancements in Research & Technology, Volume 1, Issue 4, September-2012.

[6]. Barun Biswas, Krishnendu Basuli, Samar Sen Sarma, "*On a key exchange technique, avoiding Man in the-middle Attack*", Journal of Global Research in Computer Science Volume 3, September 2012.

[7]. C. Krishna Kumar1, G. Jai Arul Jose1, C. Sajeev1 and C. Suyambulingom2, *"Safety measures against Man-in-Middle Attack in Key-Exchange"*, ARPN Journal of Engineering and Applied Sciences, vol. 7, no. 2, February 2012.

[8]. Shilpi Gupta and Jaya Sharma, *"A Hybrid Encryption Algorithm based on RSA and Diffie-Hellman"*, 2012 IEEE International Conference on Computational Intelligence and Computing Research.

[9]. Mahmood Khalel Ibrahe,*" Modification of Diffie–Hellman Key Exchange Algorithm for Zero Knowledge Proof"*, International Conference on Future Communication Networks, 2012.

[10]. Sunita, Neeraj Goyat , Annu Malik ,"*Review of Diffie–Hellman key Exchange",* International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 7, July 2013.

[11]. Jagmohan Tanti1 and R Thangadurai, *"Distribution of residues and primitive roots* ", Proc. Indian Acad. Sci. (Math. Sci.) Vol. 123, No. 2, May 2013, pp. 203–211._c Indian Academy of Sciences.

[12]. Shahab Mirzadeh, Haitham Cruickshank, Member, IEEE, and Rahim Tafazolli, Senior Member, IEEE, *"Secure Device Pairing"* , IEEE , vol. 16, 2014.

[13]. Shyam Deshmukh, Prof.Rahul Patil," *Hybrid cryptography technique using modified Diffie-Hellman and RSA",* International Journal of Computer Science and Information Technologies, Vol. 5 (6), 2014.

[14]. Akta lamba, Lalit Garg,"*Enhanced Diffie Hellman Algorithm*", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 6, June 2014.

[15]. Diffie, W., & Hellman, M. E. (1976), *"New directions in cryptography"*, IEEE Transaction on Information Theory, 22(6), 644–654 .

[16]. Rohini, Er.Meenakshi Sharma,*"Enhancing the Diffie- Hellman Algorithm*", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4.

[17]. Ritu Makani,Yogesh Chaba, *"Key Management Based Multilevel Security using Digital Signature and Encryption Technique"*, International Research Journal of Computer Science Volume 1,issue 3,2014.