Modification of Diffie-Hellman Algorithm to Provide More Secure Key Exchange

Parth Sehgal[1],Nikita Agarwal[2], Sreejita Dutta[3] ,P.M.Durai Raj Vincent [4]
[1,2,3]IIIrd B.Tech(IT),SITE, VIT University
[4] Assistant Professor(Senior), SITE, VIT University.
Parth270592@yahoo.co.in, agarwal.nikita06@gmail.com, sreejita.dutta@gmail.com

***Abstract*-- Diffie-Hellman algorithm is one of the first schemes proposed for the exchange of keys required in asymmetric encryption. It was developed by Whitfield Diffie and Martin Hellman in 1976. This algorithm removes the need of transferring keys between two communicating parties. It enables each party to generate a shared secret key for encryption and decryption of data. The security of this algorithm cannot be compromised because several security protocols and services depend upon Diffie-Hellman key exchange for reliable communication.**

In this paper, we have used a random parameter to make this algorithm more efficient. The random parameter generates new shared keys for each message that is exchanged between sender and receiver. So, different ciphertext will be produced each time even for the same message. Thus, systems using this scheme will become more tolerant to various attacks.

I. INTRODUCTION

Cryptography originated many years ago. During its early stages, two parties had to rely on a secure channel for exchanging a secret key which was used both for encryption and decryption. This scheme is known as private or symmetric key cryptography [6]. However there were faults in the security of such a scheme. To ensure complete security, the key should be at least as long as the message as proved by Claude Shannon in the 1940s. Also, a secure channel is not always available which is why we need such an encryption scheme in the first place.

These drawbacks can be overcome by using public key cryptography [5]. This scheme enables two communicating parties to agree upon a shared secret key without exchanging it over the communication channel. Instead it is derived from parameters all of which are not publicly known.

In 1976, Whitfield Diffie and Martin Hellman who were influenced by the work of Ralph Merkle on public key distribution, proposed an algorithm for key exchange which uses exponentiation in a finite field [1]. Today, Diffie Hellman is used in a variety of protocols and services. It is used in interactive transactions, rather than a batch transfer from a sender to a receiver. The algorithm is used when data is encrypted on the Web using either SSL or TLS and in VPN. So its security is of utmost importance. However, like other security algorithm it is vulnerable to various attacks like plaintext attacks, man-in the middle attacks etc. So we propose a modification of the original algorithm so as make it more tolerant and secure by using a random parameter.

Plaintext attacks are one of the most commonly used cryptanalysis methods. Samples of both the plaintext and cipher text are available to the attacker. The attacker can deduce more crucial information such as secret keys and code books from such a collection of known plaintext and cipher text. Due to the random parameter introduced in the proposed algorithm, the possibility of such a known plaintext attack is greatly reduced [8].

II. THE EXISTING DIFFIE-HELLMAN ALGORITHM

The basic version of the Diffie Hellman algorithm starts with selecting a prime number 'q' and primitive root 'a' where a<q. The powers of 'a' generates all integers from 1 to q-1, i.e. a mod q, a^2mod q, a^3 mod q,…… a^{q-1} mod q generate all integers from 1 to p-1.

In this algorithm, first user selects a random natural number 'i' as its private key. The public key for the same user is calculated as a^i mod q. Similarly the second user selects its private key as 'j'. The second user's public is generated in the same way as above. The two users then exchange their public keys. Each user uses its own private key and the other user's public key to compute the shared secret key. Using this secret key, encryption/decryption takes place.

Algorithm:

1. Select prime number 'q' and its primitive root 'a' (1<= a <=q).
2. User 1:
   - Selects pr(1)=i
   - pu(1)= a^i mod q
3. User 2:

- Selects pr(2)=j
- pu(2)= a^j mod q

4. Both parties exchange public keys.
5. User1:
   - Shared secret key, K= $pu(2)^{pr(1)}$
     = a^{ji} mod q
6. User 2:
   - Shared secret key, K=$pu(1)^{pr(2)}$
     = ai^j mod q

Only the communicating parties know the private keys i, j as they are not transmitted and cannot be intercepted. So only the communicating parties can calculate the shared secret key. However there is still no way of authenticating the communicating parties as their identity is not linked to the keys that they share. So the algorithm is still prone to man-in-the-middle attacks. To avoid such attacks, public key certificates and digital signatures can be used, as described in the Authenticated Diffie Hellman Key Agreement Protocol [2].

Also, as the shared key remains constant for a session, the same message when encrypted multiple times will give the same cryptic text each time. So, frequently occurring patterns in ciphertext can be used to find relationships between the plaintext and the ciphertext. So, to avoid known plaintext attacks, we introduce a random factor in the key agreement protocol so that a different ciphertext is generated even for the same plaintext each time it is encrypted.

III. MODIFICATION TO THE DIFFIE HELLMAN ALGORITHM

Here too, first we select a prime number ‘q’ and its primitive root ‘a’, where a<q.

The first user then selects a random natural number ‘i’ as its private key. Its public key is calculated as a^i mod q. Similarly, the second user selects a random natural number ‘j’ as its private key. Its public key is calculated as a^j mod q. The public keys are then exchanged over a public channel. Both users now selects a random integer ‘t’,’s’, 0<t,s<q.

The random integer ‘t,s’ is then disguised and exchanged between the two users. The other user can extract the value of the random integer from the message as they have a knowledge of the private as well as public keys.

The shared secret key is calculated by the first user using its own private key, the second user’s public key and the random integers ‘t,s’, through exponentiation in a finite field. This secret key is used to encrypt and decrypt the message.

Algorithm:

1. Select prime number ‘q’ and its primitive root ‘a’ (1<= a <=q).
2. User 1:
   - Selects pr(1)=i, 0<i<q
   - pu(1)= a^i mod q
3. User 2:
   - Selects pr(2)=j, 0<j<q
   - pu(2)= a^j mod q
4. Both parties exchange public keys.
5. User 1:
   - Selects random integer ‘t’, 0<t<q
   - Calculate x= $pu(1)^{pr(2)}$
     = a^{ij} mod q
   - Send $t.a^{ij}$to User 2.
6. User 2:
   - Selects random integer ‘s’, 0<s<q
   - Calculate y= $pu(1)^{pr(2)}$
     = a^{ij} mod q
   - Send $s.a^{ij}$to User 1.
   - Extract ‘t’ as t=$(t.a^{ij})/y$
7. User 1:
   - Extract ‘s’ as s=$(s.a^{ij})/x$
   - Shared secret key,
     K= $pu(2)^{t.s.pr(1)}$

= $a^{ts.ji}$ mod q

- Encrypt the plaintext as:

  C=E(M,K).

8. User 2:

- Shared secret key,

  $K=x^{t.s}$

  $= a^{t.s.ji}$ mod q

- Decrypt the cipher text as:

  M=D(C,K)

The random factor in the shared secret key, which is different for each message, improvesthe security ofthe Diffie Hellman algorithm against known plaintext attacks. The same block of text, is encrypted with a different key each time to generate a new ciphertext each time. So even if an attacker intercepts the messages being transmitted, he cannot map it to a known set of plaintext and ciphertext. Also, the keys cannot be derived from such a message.

IV. ANALYSIS AND RESULTS

In a known plaintext attack, a set of plaintext and their corresponding ciphertext is available to the attacker [7]. Using such a set, the attacker can easily find the plaintext if the corresponding ciphertext is present in the set. Furthermore, an analysis of such a set can determine the shared secret key used in the communication and all future messages can then be easily decrypted.

This kind of an attack is quite probable if the original Diffie Hellman key exchange protocol is used. However by introducing a random factor in the calculation of the shared secret key, we can ensure that even if such a set of plaintext and corresponding ciphertext becomes available to the attacker, no two messages, or two instances of the same message will have the same corresponding ciphertext. So, the messages cannot be mapped to such a set to acquire the plaintext. Known plaintext attacks can be avoided thus.
To verify the feasibility of the modified Diffie-Hellman algorithm, we divided the process of converting a plaintext to ciphertext into three parts, key exchange, encryption and decryption and observed the time taken by each of these parts to execute individually. The same was done for the original Diffie-Hellman algorithm as well. We then plotted a graph indicating the total runtime of the existing as well as the proposed Diffie-Hellman algorithm for different data sizes.

TABLE I: RUNTIME FOR BASIC DIFFIE-HELLMAN ALGORITHM FOR DIFFERENT DATA SIZES.

message size	**2**	**3**	**4**	**5**
key exchange (in ns)	3710170	3609236	4542445	3694773
encryption(in ns)	872478	679592	760852	729203
decryption(in ns)	3482037219	4054036323	4696896031	3963999165
total time(in ns)	3486619867	4058325151	4702199328	3968423141

TABLE II: RUNTIME FOR MODIFIED DIFFIE-HELLMAN ALGORITHM FOR DIFFERENT DATA SIZES.

message size	**2**	**3**	**4**	**5**
key exchange (in ns)	3815381	4375648	4078835	4065148
encryption(in ns)	1885236	2256040	2445504	2521632
decryption(in ns)	5604600268	4910492317	5041517974	4511353761
total time(in ns)	5610300885	4917124005	5048042313	4517940541

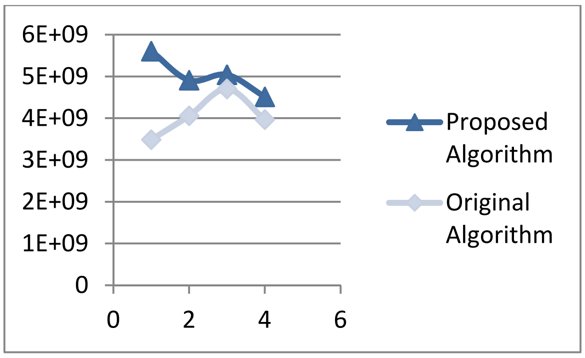

Fig 1. Comparison of runtimes for the proposed algorithm and the original algorithm.

In Figure 1, the x-axis represents the size of the message to be encrypted and the y-axis represents the total time taken (in nanoseconds) for the process, which includes key generation, encryption and decryption. It is observed that the execution time of the modified algorithm is greater than that of the existing algorithm.

However, the difference between the two runtimes is very small and is observed to be about 969460064 nanoseconds.
This increase in runtime in case of the proposed algorithm is a small tradeoff for improved security of the original algorithm. So we can say that it is feasible to introduce the random parameters in the existing Diffie-Hellman algorithm so as to make it less vulnerable to known plaintext attacks, thereby improving the security of the algorithm.

V. CONCLUSION

The basic version of Diffie Hellman algorithm faces multiple security threats. The security of the algorithm depends on the difficulty of solving discrete logarithms and of the integer factorization problem. The security also depends on the bit length of the keys used [3][4].

In this paper, we have proposed an improvement over the Diffie Hellman Algorithm where a random factor is introduced in the secret key for each message that is encrypted. So, a new ciphertext is generated even for the same plaintext each time. This reduces a possibility of a known plaintext attack as explained above.

This security improvement is beneficial because Diffie Hellman Algorithm is the basis of several security standards and services on the internet, and if the security of the Diffie Hellman algorithm is compromised, such systems will collapse.

VI. REFERENCES

[1] Diffie W., Hellman M., 1976. “New directions in cryptography”, IEEE Transactions on Information Theory, volume 22, pages 644-654.
[2] Diffie, W.; van Oorschot, P. C.; Wiener, M. J. (1992), "Authentication and Authenticated Key Exchanges", Designs, Codes and Cryptography (Kluwer Academic Publishers) 2: 107–125.
[3] Maurer U.M., 1994. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms, 271–281.
[4] Francois J., Raymond A., 1998. Security Issues in the Diffie-Hellman Key Agreement Protocol, IEEE Trans. on Information Theory, pages 1–17.
[5] Martin E. Hellman May 2002. An Overview of Public Key Cryptography, IEEE Communications Magazine, pages:42–49.
[6] Bellare, M., Canetti, R., and Krawczyk, H. Modular approach to the design and analysis of key exchange protocols. In Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC-98) (New York, May 23–26 1998),
ACM Press, pages. 419–428.
[7] Diffie, Whitfield, and Martin E. Hellman, 1977. "Special feature exhaustive cryptanalysis of the NBS data encryption standard." Computer 10.6, pages : 74-84.
[8] Diffie, W., & Hellman, M. E. (1979). Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE*, *67*(3), pages 397-427.