Research on Diffie-Hellman Key Exchange Protocol

Nan Li
Information Engineering Teaching and research section
The People's Armed Police Force Academy of China
Langfang Hebei 065000, China
linan@wjxy.edu.cn

***Abstract*—The purpose of the Diffie-Hellman protocol is to enable two users to exchange a secret key securely that can then be used for subsequent encryption of messages. The protocol itself is limited to exchange of the keys. But because of having no entity authentication mechanism, Diffie-Hellman protocol is easily attacked by the man-in-the-middle attack and impersonation attack in practice. In this paper, we compare the computational efficiency of various authentication methods. Finally an improved key exchange schema based on hash function is given, which improves the security and practicality of Diffie-Hellman protocol.**

Keywords-authentication mechanism; Diffie-Hellman protocol; key exchange

I. INTRODUCTION

Diffie-Hellman key exchange (D-H) is a cryptographic protocol that allows two parties that have no prior knowledge of each other to establish together a shared secret key over an insecure communications channel. Then they use this key to encrypt subsequent communications using a symmetric-key cipher. The scheme was first published publicly by Whitfield Diffie and Martin Hellman in 1976, Diffie-Hellman key agreement itself is an anonymous (non-authenticated) key-agreement protocol, it provides the basis for a variety of authenticated protocols, and is used to provide perfect forward secrecy in Transport Layer Security's short-lived modes as in [1]. In the original description papers, the Diffie-Hellman exchange by itself does not provide authentication of the communicating parties and is thus susceptible to a man-in-the-middle attack. An attacking person in the middle may establish two different Diffie-Hellman key exchanges, with the two members of the party "A" and "B", appearing as "A" to "B", and vice versa, allowing the attacker to decrypt (and read or store) then re-encrypt the messages passed between them. A method to authenticate the communicating parties to each other is generally needed to prevent this type of attack.

II. DIFFIE-HELLMAN KEY EXCHANGE PROTOCOL

The Diffie-Hellman algorithm depends for its difficulty of computing discrete logarithms. Briefly, we can define the discrete logarithm in the following way. First, we define a primitive root of a prime number p as one whose powers gen all the integers from 1 to p-1. That is, if a is a primitive root of the prime number p, then the numbers

$$a \bmod p, a^2 \bmod p, ..., a^{p-1} \bmod p$$

are distinct and consist of the integers from 1 through p-1 in some permutation.

For any integer b less than p and a primitive root a of prime number p, one can find a unique exponent i such that

$$b = a^i \bmod p \quad (1)$$

where $0 \le i \le (p\text{-}1)$ The exponent i is referred to as the discrete logarithm, or index, of b for the base a, mod p. This value is denoted as $\text{ind}_{a,\,p}$ (b). As in [2], for this scheme, there are two publicly known numbers: a prime number q and an integer α that is a primitive root of q. Suppose the users A and B wish to exchange a key. User A selects a random number $X_A<q$ and calculates the public key $Y_A=\alpha^{X_A} \bmod q$. Similarly, user B independently selects a random number $X_B<q$, and calculates the public key $Y_B=\alpha^{X_B} \bmod q$. Each side keeps the X value private and makes the Y value available publicly to the other side. User A calculates the key as $K = (Y_B)^{X_A} \bmod q$ and user B calculates the key as $K = (Y_A)^{X_B} \bmod q$. These two calculations produce the same results:

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod q \\ &= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \\ &= \alpha^{X_B X_A} \bmod q \\ &= (\alpha^{X_A})^{X_B} \bmod q \\ &= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\ &= (Y_A)^{X_B} \bmod q \end{aligned} \quad (2)$$

Thus, the two sides have exchanged a secret key. Furthermore, because X_A and X_B are private an opponent only has the following ingredients to work with: q, α, Y_A and Y_B. Thus, the opponent is forced to take a discrete logarithm to determine the key. For example, attacking the secret key of user B, the opponent must compute

$$X_B = \text{ind}_{\alpha,q}(Y_B) \quad (3)$$

The opponent can then calculate the key K in the same manner as user B calculates it. The Diffie-Hellman key exchange algorithm's security depends on this fact: while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithm. For large primes, the latter task is considered infeasible. For example, although the attacker knows that p, g, Y_A, Y_B, but he can not know the X_A, X_B, so he can not know the secret key K_{AB}. To know the X_A, X_B, the attacker must demand discrete logarithm equation $\alpha = b^x \bmod p$ in a finite field GF(p). But it is very difficult to calculate X even that the attacker knows the α, b and p. The computational complexity is

$$L(p) = e((\ln p)1/3 \ln(\ln p)2/3 \quad (4)$$

So, using the Diffie-Hellman algorithm can establish a shared secret that can be used for secret communications by

978-1-4244-6349-7/10/$26.00 ©2010 IEEE

exchanging data over a public network. Fig.1 shows a simple protocol that makes use of the Diffie-Hellman calculation. Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection. User A can generate a one–time private key X_A, calculate Y_A, and send that to user B. User B responds by generating a private value X_B, calculating Y_B, and sending Y_B to user A. Both users can now calculate the key. The necessary public values q and α would need to be known ahead of time. Alternatively, user A could pick values for q and α and include those in the first message.

III. DIFFIE-HELLMAN PROTOCOL ANALYSIS

A. *Diffie-Hellman's Features*

As in [3], the Diffie-Hellman algorithm has two attractive features:

- Secret keys are created only when needed. There is no need to store secret keys for a long period of time, exposing them to increased vulnerability.
- The exchange requires no preexisting infrastructure other than an agreement on the global parameters.

However, there are a number of weaknesses to Diffie-Hellman algorithm:

- It does not provide any information about the identities of both parties. So it is vulnerable to impersonation attack.
- It is computationally intensive. As a result, it is vulnerable to a clogging attack, in which an opponent requests a high number of keys. The victim spends considerable computing resources doing useless modular exponentiation rather than real work.
- It can not prevent replay attack.
- It is subject to a man-in-the-middle attack, in which a third party C impersonates while communicating with A and impersonates A while communicating with B. Both A and B end up negotiating a key with C, which can then listen to and pass on traffic.

B. *Man-in-the-middle Attack*

As in [4], the man-in-the-middle attack proceeds as follows:

step1: A → C (B): $\alpha^{X_A} \bmod q$
step2: C (A) → B: $\alpha^{X_C} \bmod q$
step3: B → C (A): $\alpha^{X_B} \bmod q$
step4: C (B) → A: $\alpha^{X_C} \bmod q$

User A generates a one–time private key X_A, calculates Y_A, and sends his public key Y_A in a message addressed to user B. The enemy C intercepts A's message. C saves A's public key (Y_A) and generates a one–time private key X_C, calculates Y_C, and sends his public key Y_C in a message to user B. This message to B has A's User ID but C's public key Y_C. This message is send in such a way that it appears as though it was send from A's host system. B receive C's message and stores C's public key with A's User ID.

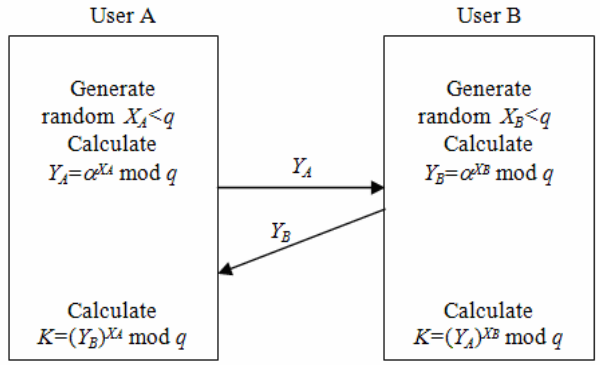

Figure 1. Diffie-Hellman key exchange

Similarly, C sends a message to A with C's public key (Y_C), purporting to come from B. A calculates a secret key K_1 ($(Y_C)^{X_A} \bmod q$) based on X_A and Y_C. C calculates K_1 ($(Y_A)^{X_C} \bmod q$) using X_C and Y_A. So K_1 is shared by A and C. B calculates a secret key K_2 ($(Y_C)^{X_B} \bmod q$) based on X_B and Y_C. C calculates ($(Y_B)^{X_C} \bmod q$) using X_C and Y_B. So K_2 is shared by B and C. From now on user A thinks K_1 is shared with B, user B thinks K_2 is shared with A, but no key is shared by A and B actually. So C is able to relay massages from A to B and from B to A.

C. *Impersonation Attack*

The impersonation attack proceeds as follows:

step1: A → C(B): $\alpha^{X_A} \bmod q$
step2: C(B) → A: $\alpha^{X_C} \bmod q$

User A sends his public key Y_A in a message addressed to user B, but the message is intercepted by enemy C. C saves A's public key and sends a message to A back which includes B's User ID but C's public key Y_C. In this attack, B is simply not involved in implementation of the agreement. The attack result is C and A shared key ($(Y_C)^{X_A} \bmod q$), while the A thinks the key is t shared with B.

From the analysis of Diffie-Hellman protocol we can see the fundamental reason that the protocol subject to the above attacks is that there is no authentication between the two parties [5].

IV. THE IMPROVED PROTOCOL

Usually there are three different authentication methods used with Diffie-Hellman protocol [6]:

1) Digital signatures: The exchange is authenticated by signing a mutually obtainable hash; each party encrypts the hash with its private key. The hash is generated over important parameters, such as user ID and nonce.

2) Public-key encryption: The exchange is authenticated by encrypting parameters such an ID and nonce with the sender's private key.

3) Symmetric-key encryption: A key derived by some out-of-band mechanism can be used to authenticate the exchange by symmetric encryption of exchange parameters.

In order to compare the computational efficiency of various authentication methods, we selected a number of

widely used and recognized security algorithms to be tested. We select MD5 (Message-Digest Algorithm 5) and SHA-1 (Secure Hash Algorithm 1) as the hash algorithm, AES (Advanced Encryption Standard) and DES (Data Encryption Standard) as the symmetric encryption, ECC (Elliptic Curve Cryptography) and RSA as the public key encryption. Test environment is as follows: Intel Pentium4 2400 MHZ, 256MB RAM. The test results are shown in Fig.2, which expresses each algorithm's number of operations per second while the 64-byte data blocks as input. As the computing speed of various algorithms vary widely, so we use logarithmic coordinate in Fig.2. Through the comparison of variety of encryption algorithms' computing speed, we can see the computing speed of either ECC or RSA is far lower than the hash algorithms and symmetric key encryption. Generally speaking, the computing speed of hash algorithm is the symmetric key encryption's an order of magnitude, is signature encryption's about four orders of magnitude. From the intuitive point of view, if uses 10 hours to generate a digital signature, while just 2.2 seconds to carry out a hash calculation. Moreover, whether digital signature algorithm or public key encryption requires a dedicated PKI (public key infrastructure), which is the main reason a lot of authentication mechanisms use hash calculation and avoid the public key encryption.

Therefore, we propose an improved Diffie-Hellman key exchange protocol based on hash function. In this protocol, A and B are the two sides. The key exchange protocol proceeds as follows:

Srep1: A→AS, $ID_A||ID_B$

Step2: AS→A, $N_1 \oplus P_A$

Step3: AS → B, $N_1 \oplus P_B$

Srep4: A→B, $Y_A||H(Y_A||N_1)$

Step5: B→A, $Y_B||H(Y_B||f(N_1))$

Step6: A → B, $H(N_1)$

Step7: A calculates $K = (Y_B)^{X_A}$

B calculates $K = (Y_A)^{X_B}$

Where

AS = authentication server

ID_A = identifier of user on A

ID_B = identifier of user on B

P_A = password of user on A

P_B = password of user on B

N_1 = one-time random number

$\oplus$ = XOR

$||$ = concatenation

H = hash function such as MD5 or SHA-1

$Y_A = \alpha^{X_A} \bmod q$

$Y_B = \alpha^{X_B} \bmod q$

f = simple transformation (addition, subtraction or shift operation)

As shown in Fig.3, user A sends a request message to the AS that includes the ID_A and ID_B. Then the AS responds this message by sending $N_1 \oplus P_A$ to A and $N_1 \oplus P_B$ to B according to ID_B. Being a simple encryption, XOR can encrypt N_1. User A can calculate $N_1 \oplus P_A \oplus P_A$ to receive N_1 and user B also calculate $N_1 \oplus P_B \oplus P_B$ to receive N_1. Now, N_1 is shared by A and B. And then, user A sends $Y_A||H(Y_A||N_1)$ to B, B calculates $H'(Y_A||N_1)$ by Y_A from A and N_1 from AS. If $H'(Y_A||N_1)$ is equal to $H(Y_A||N_1)$, B believes this message sent by A, or interrupts this communication. Similarly, A calculates $H'(Y_B||f(N_1))$ by Y_B from A and N_1 from AS. If $H'(Y_B||f(N_1))$ is equal to $H(Y_B||f(N_1))$, A believes this message sent by B and calculates the $K = (Y_A)^{X_B} \bmod q$, or interrupts this communication. After A sends a massage $H(N_1)$ as a confirmation signal to B, A and B calculate the $K = (Y_A)^{X_B} \bmod q = (Y_B)^{X_A} \bmod q$.

V. THE ANALYSIS OF THE IMPROVED PROTOCOL

A. *Analysis of Security*

- The introduction of random value N1 assures that the response is fresh and has not been replayed by an opponent.
- N1 is the secret information only between user A and B, so it is difficult for others to know. Therefore, the use of N1 can be effective on the identity authentication of both sides and resisting man-in-the-middle attack and impersonation attack.
- That the user B calculates the key after the confirmation message sent back from user A can resist clogging attack efficiently.

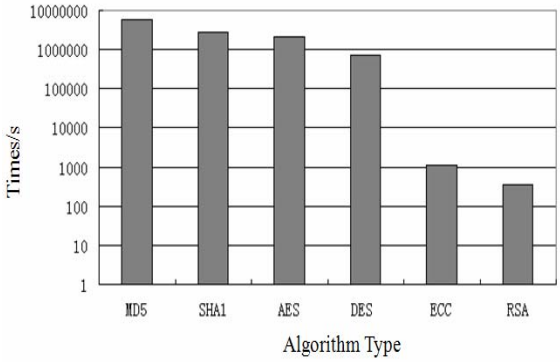

Figure 2. Computing speed of authentication method

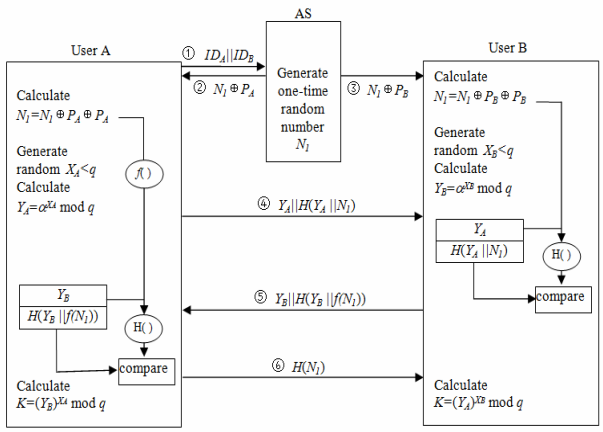

Figure 3. Improved Diffie-Hellman key exchange protoco

B. Analysis of computing speed

In the key exchange process, it is small number of bytes, generally a few bytes to a few dozen bytes, to be need for the safe handling， such as N_1 is less then 10000D and tends to be used by two or three bytes, Y_A and Y_B use 128 bytes in generally. For some security algorithms, the average speed of the safe handling of large random data and handling of small random data may be very different.

By simulating the various key exchange processes based hash function, public-key encryption and Symmetric-key encryption within the LAN, we can get the results shown in Table.I. It can be seen from Table.I that hash function's efficiency is more effective than the symmetric-key encryption algorithm's for small message, such as MD5 algorithm's speed is about 5 times faster than the DES algorithm's speed and SHA-1's speed is more twice than the DES's speed. For example, when 64 bytes are input, 2010,000 times per second is completed by MD5 operations, 900,000 times per second completed by SHA-1 operation, while for symmetric algorithms, the completion of 300,000 times per second only by DES operations. So we can see that when small pieces of data are encrypted using symmetric-key algorithm, the time used to transform the key will have some big impact on calculating speed. Therefore, for small data encryption, the encryption performance depends on the key-transforming time and the encryption time. The Hash calculation does not depend on the key, so the efficiency of symmetric-key encryption will be reduced relatively for small data but hash algorithm will not. In addition we can see, the calculating speed of MD5 algorithm is twice faster than the calculating speed of SHA-1 algorithm or so, but the SHA-1 is considered more secure. Although SHA-1 algorithm's computing speed is slower than MD5 algorithm's, but far faster than the DES encryption and the RSA encryption. Therefore, in the key exchange protocol, we recommend selecting the SHA-1 algorithm as hash function.

VI. CONCLUSION

By analyzing the security of the Diffie-Hellman protocol, this paper presents an improved key exchange protocol based on random number sequence. This protocol using a hash function to achieve authentication is a relatively simple, economical and practical programs without additional public key infrastructure as a support. Because of including Authentication mechanism, the improved Diffie-Hellman key exchange protocol can resist replay attack, impersonation attack and man-in-the-middle attack. The simulation results in the LAN prove that authentication using hash function has the less computing quantity and the faster computing speed than the other public key and symmetric key encryption algorithm. It has a high practical value in building a secure communication channel of symmetric key.

REFERENCES

[1] Ian F. Blake and Theo Garefalakis, On the complexity of the Discrete Logarithm and Di_e-Hellman problems, Journal of Complexity 20 (2004), 148–170.

[2] Willian Stallings, Network Security Essentials: Applications and Standards, 2nd ed, Beijng: qinghua press, 2004.1, pp.75–77.

[3] Li Xin, An Improvement of Diffie-Hellman Protocol, Network & Computer Security, 2007,12, pp. 22–23.

[4] Jung Hee Cheon and Byungheup Jun, A polynomial time algoritm for the braid Di_e-Hellman conjugacy problem, Advances in cryptography – CRYPTO 2003, Lecture Notes in Computer Science, no. 2729, Springer, Berlin, 2003, pp. 212–225.

[5] Steven Galbraith and Victor Rotger, Easy decision Di_e-Hellman groups, LMS Journal of Computation and Mathematics 7 (2004), 201–218.

[6] Ueli Maurer and Stefan Wolf, On the complexity of breaking the De_e-Hellman protocol, Advances in Cryptology - CRYPTO'96, Lecture Notes in Computer Science, vol. 1109, Springer-Verlag, 1996, pp. 268–282.

TABLE I. PERFORMANCE COMPARISON OF VARIOUS ALGORITHMS

Encryption	**Computing speed of different bytes(1000 times/s)**				**Average time of key exchange (ms)**
	10	***64***	***128***	***150***	
MD5	3360	2020	1920	1370	0.01
SHA-1	1860	900	910	720	0.21
DES	720	300	220	150	0.79