An Efficient Session Key Generation Protocol

Chin-Chen Chang, Chi-Yien Chung

Department of Computer Science and Information Engineering

Nation Chung Cheng University

Chiayi, Taiwan 621, R.O.C.

E-mail:{ccc, jjy89}@cs.ccu.edu.tw

FAX:886-5-2720859

***Abstract*-Key exchange is an important subject in cryptosystem, which has marched the world into the complicated application of communication technology nowadays. A common cryptographic technique is used to encrypt each individual's conversation with a separate key, which is so-called session key. The lifespan of a session is the period of a particular communication session. Instead of depending on discrete logarithm like many previously proposed schemes, we use public key cryptography here. In this paper, we will combine key exchange with digital signature. Adopting the ring topology and then co-operating with the RSA algorithm to encrypt secret random numbers, every one gets all the random numbers after several turns, and then the session key is generated. Because of the presence of a trusted third party, our proposed protocol can withstand the man-in-the-middle attack as well.**

Keywords: Key exchange, session key, public key cryptography.

1 Introduction

Nowadays, network communication is becoming more and more popular. When we need to send secret messages through the network system, cryptography will come in and play an important role in security. If a group of people want to hold a very secure conference through a network communication system, how do they converse with one another securely? A simple way is to encrypt each individual's conversation with a session key[3]. For the sake of security, the session key only exits for the duration of the communication. The only rule is, all of the players in the session have to agree on the same key, and no one out of the communication could impersonate a player in the session and trick legitimate users into encrypting messages with her/his phony key. There are several algorithms [1,2,4,5,6,7] and topologies to meet this requirement for the time being.

Unlike the scheme based on discrete logarithm, we use public key [10] cryptography instead here. In this paper, we shall combine key exchange with digital signature [9,10], the latter of which can withstand the man-in-the-middle attack as well. We assume that there is a trusted third party, say Trent, who would sign the public keys of the players in the session, and every one who receives the key could verify Trent's signature to make sure the public key indeed belongs to this gentleman. The key exchange protocol can then proceed safely. In our protocol, we adopt the ring topology[8] and co-operate with the RSA algorithm to encrypt a set of chosen secret random numbers. After several turns, every one gets all the random numbers, and then they will have the ability to compute the session key. The announced public key has been verified by Trent, and therefore there is no way to impersonate. As a result, since the session key remains a secret in the communication period and could not be disclosed, the ciphertext would not be understood by any other people out of this session. We can see that this session key plays an important role in network communication and security defense.

This paper is organized as follows. We shall have a review of previous schemes in Section 2. After that, a new efficient session key generation scheme will be presented in Section 3. Finally, the conclusion will be given in Section 4.

2 A Review of Past Research works

2.1 Diffie-Hellman with Three or More Parties

The security of Diffie-Hellman public key algorithm is based on the difficulty of calculating logarithms in a finite field, as compared with the ease of calculating exponentiation in the same field. The Diffie-Hellman [2] key-exchange protocol can also be extended to work with more than three

people. Here we shall use an example to illustrate their protocol. In this example [12], Alice, Bob, and Carol together generate a secret key.

(1) Alice chooses a large random integer x and sends Bob X, where

$$X = g^x \mod n.$$

(2) Bob chooses a large random integer y and sends Carol Y, where

$$Y = g^y \mod n.$$

(3) Carol chooses a large random integer z and sends Alice Z, where

$$Z = g^z \mod n.$$

(4) Alice sends Bob Z', where

$$Z' = Z^x \mod n.$$

(5) Bob sends Carol X', where

$$X' = X^y \mod n.$$

(6) Carol sends Alice Y', where

$$Y' = Y^z \mod n.$$

(7) Alice computes k, where

$$k = Y'^x \mod n.$$

(8) Bob computes

$$k = Z'^y \mod n.$$

(9) Carol computes

$$k = X'^z \mod n.$$

The secret key, k, is equal to $g^{xyz} \mod n$. No one out of the communication could have the same key. Of course, this protocol could also be extended to more than three people as well in the same way.

2.2 Diffie-Hellman with Another Topology

As illustrated in the above example, we find that these three vertices pass X, Y, Z, X', Y', and Z' cyclically and finally get the session key k. Recently, Steiner and Tsudih [11] proposed another protocol, which implements the same achievement but uses different topology, for multiparty authentication services. Here is how this protocol works.

Assume there are three players: P_1, P_2, P_3, and a group controller in this communication session who want to generate a common session key together. At the beginning, any two of them have a key. The name of the key k_{ij} means that it is key shared between P_i and P_j. For example, P_1 has the key k_{12} shared with P_2 and the key k_{13} shared with P_3 and the key k_{14} shared with the group controller.

(1) P_1 sends $g^{r_1 k_{12}}, g^{r_1 k_{13}}, g^{r_1 k_{14}}$ to P_2, where r_1 is chose by P_1.

(2) P_2 sends

$g^{r_1 k_{12}}, g^{r_2 k_{21}}, g^{r_1 r_2 k_{13} k_{23}}, g^{r_1 r_2 k_{14} k_{24}}$ to P_3.

(3) P_3 sends

$g^{r_1 r_2 k_{13} k_{23}}, g^{r_2 r_3 k_{21} k_{31}}, g^{r_1 r_3 k_{12} k_{32}}, g^{r_1 r_2 r_3 k_{14} k_{24} k_{34}}$ to group controller.

(4) Group controller sends $g^{r_2 r_3 r_4 k_{12} k_{13} k_{14}}$ to P_1 and then P_1 could compute the session key

$$k = (g^{r_2 r_3 r_4 k_{12} k_{13} k_{14}})^{r_1 k_{12}^{-1} k_{13}^{-1} k_{14}^{-1}} = g^{r_1 r_2 r_3 r_4}.$$

(5) Group controller sends $g^{r_1 r_3 r_4 k_{21} k_{23} k_{24}}$ to P_2 and then P_2 could compute the session key

$$k = (g^{r_1 r_3 r_4 k_{21} k_{23} k_{24}})^{r_2 k_{21}^{-1} k_{23}^{-1} k_{24}^{-1}} = g^{r_1 r_2 r_3 r_4}.$$

(6) The group controller sends $g^{r_1 r_2 r_4 k_{31} k_{32} k_{34}}$ to P_3 and then P_3 could compute the session key

$$k = (g^{r_1 r_2 r_4 k_{31} k_{32} k_{34}})^{r_3 k_{31}^{-1} k_{32}^{-1} k_{34}^{-1}} = g^{r_1 r_2 r_3 r_4}.$$

The advantage of this protocol is that it could dynamically change the group. It is easy to add or delete a member in this communication group. But the disadvantage is that it could not confirm the identity of the user.

2.3 Extended Diffie-Hellman

In [13], Hughes presented an encrypted key transmission protocol. This different kind of edition, which is also based on Diffie-Hellman, allows the chairperson to generate a key and send it to all the players in the communication session.

Assume that there are three people Alice, Bob, and Carol in this session, and Alice wants to generate a key and distribute it to Bob and Carol.

(1) Alice chooses a large random integer x and generates

$$k = g^x \mod n.$$

(2) Bob chooses a large random integer y and sends Alice

$$Y = g^y \mod n.$$

(3) Carol chooses a large random integer z and sends Alice

$$Z = g^z \mod n.$$

(4) Alice sends Bob

$$X = Y^x \mod n.$$

(5) Alice sends Carol

$$X' = Z^x \mod n.$$

(6) Bob computes

$$a = y^{-1},$$
$$k' = X^a \mod n.$$

(7) Carol computes

$$b = z^{-1},$$
$$k'' = X^b \mod n.$$

If everything goes right, they could have k=k'=k''. Then they could communicate with this session

key if they wish to.

The advantage of this protocol over the previous two is that k could be pre-computed before any interaction, and the one who generates the key could encrypt a message using k prior to negotiating with the others.

3 The Proposed Protocol

Our proposed protocol enjoys the same flavor and simplicity as the Diffie-Hellman protocol; i.e., each player has his own secret value, and the session key is generated by Exclusive-OR these values.

We adopt the ring topology in this paper where all players pass the secret numbers cyclically. Assume that our computation model is composed of a set of n players $P_1,.....,P_n$, where P_1 passes the secret random number r_1 to P_2, and P_2 passes his own r_2 one to P_3 ,...., and P_n passes r_n to P_1 at round 1. Then repeating again, P_1 passes his newly received secret value r_n to P_2 , and then P_2 passes r_1 to P_3, P_3 passes r_2 to P_4 and so on. As we can expect, every one will collect all the others' secret numbers after several rounds, and then they can obtain the same session key for the communication afterwards.

However, we shall try a different way to pass the secret values among the players in the session. Here we assume that our computation model is composed of a set of n players $P_1, P_2,..., P_n$. We start by choosing a secret value individually, and then player P_i passes to P_{i+1} her/his own secret value encrypted with the public key of P_{i+1} synchronously at the first round where i is from 1 to n. Then P_{i+1} passes her/his encrypted value to P_1, which completes a cycle therefore. Every one who gets the encrypted value could decrypt and keep it using her/his own private key. The following steps are: each player P_i passes to P_{i+1} her/his newly received value encrypted with the public key of P_{i+1}; after n rounds, each one of the players has all the secret values, and the session key is the Exclusive-OR of these values.

To realize the above protocol, we make the communication topology a ring; i.e., the passing work is done cyclically. Namely, player P_1 passes the value to P_2, and P_2 does the same to P_3, and the same thing goes on and on until at last P_n passes the value to P_1 completes the first round. When n rounds are done, each player has all the others' values, so a session key can be generated after the Exclusive-OR operation.

Of course we assume that Trent, the trusted third party, is absolutely honest so that the protocol can proceed smoothly. Trent would guarantee through the certificate that the public key each player gets truly belongs to himself. The certificate includes the public key and the ownership, signed with the private key of Trent. The one who receives the certificate could verify the signature of Trent to make sure of the published public key. As a result, each player would feel comfortable using the announced public key to encrypt the secret value, and only the one who has the corresponding private key could decrypt the encrypted value.

3.1 Security Analysis

Man-In-The-Middle Attack

The man-in-the-middle attack works because the players have no way to verify that they are talking to the exact person they really wish to. If the network communication is not noticeably delayed, the two parties in conversation would not have any idea that someone sitting between them is reading all their supposedly secret messages.

While intruders cannot do any better than trying to break the whole encryption system or the cipher-text only attack, then still would much like to try this attack, because it could enable them not only to listen to the messages between two players but also to modify the messages. This attack could cause deadly damage if the messages involve military secrets or commercial information. The following is how the attack works:

Assume Alice and Bob want to communicate with each other. Suppose Eve is the intruder who attempts to eavesdrop on them.

(1) Alice sends Bob his public key for encryption and Eve intercepts this key and replaces it with his own public key instead.
(2) Bob sends Alice his public key for encryption and Eve intercepts this key and replaces it with his own public key instead.
(3) When Alice sends a message to Bob, encrypted with "Bob's" public key, Eve intercepts it. Since the message is encrypted with Eve's public key actually, Eve can decrypt it with his corresponding private key and modify that message and then encrypt this modified message with Bob's public key and send it to Bob.
(4) When Bob sends a message to Alice, encrypted with "Alice's" public key, Eve intercepts it. Since the message is encrypted with Eve's public key actually, Eve can decrypt it with his corresponding private key and modify that message and then encrypt this modified message with Alice's public key and sends it to Alice.

The major problem here is that there is no way for Alice and Bob to make sure that the public key truly belongs to the one they think. The

public key is made in public, so each player can announce her/his own. Without a trusted third party to guarantee that the public key is accurate, the announced one is not convincing.

In our proposed protocol, we have a Trent who is responsible for the endorsement of the public key. Every one registers her/his public key to Trent and Trent would identify the ownership and sign for it. Now, if there is a player sitting between Alice and Bob, she/he could no longer impersonate Alice and sends a fake public key to convince Alice that this key belongs to Bob, and vice versa. As a result, the man-in-the-middle attack becomes impossible. The only thing that the intruder could do is disturbing with no much harm done.

3.2 Remarks

Security. In public key cryptography system, there is a noticeable problem that the verification of announced public keys is a must-do. In our proposed protocol, we introduce the concept of Trent to guarantee the correctness of the public keys such that there is no way to impersonate and eavesdrop on the communications between two players; i.e., the man-in-the-middle attack is no good. Public keys cryptography could also do authentication as well, so that only the one with the corresponding private key could decrypt the message. This helps a lot in confidentiality. The only thing the intruder could do is disturbing and no secrets would be revealed. The security relies on the absolutely honest Trent. Here we assume Trent is unbreakable of course. In this case, the security is promised.

After all the secret values are collected, the session key is just the exclusive-OR of all these values. This is pretty simple and fast. Besides, adopting the ring topology would simplify the procedure of the session key generation also. Regularity is the advantage of this topology, which makes the method simple and clear. We would like to point out here that our secure protocol does not lose the efficiency and simplicity the previously known session key generation protocol provides.

4 Conclusions

A session key is used in communication among a group of people that can accelerate the speed of encryption and decryption better than the public key algorithm. Therefore, key generation is an important topic in our real life nowadays. In our proposed protocol, we concern not only about the security but also simplicity. Based on RSA cryptography instead of discrete logarithm, we achieve the goal of authentication and confidentiality, making the man-in-the-middle attack neutralized. By introducing the concept of Trent, we can solve the problem of fake public key announcement. A noticeable observation obtained in this paper is that the ring topology can simplify the whole problem and make the structure simple and regular. We have borrowed a lot of ideas from the Diffie-Hellman protocol, and we have made it more defensive. Preventing from impersonating, we could make a session key generation protocol work more correctly and efficiently. More importantly, our newly proposed protocol is easy and simple, and it enjoys the same flavor and security as Diffie-Hellman protocol does.

References

1. J. Brandt, I.B. Damgard, P. Landrock, and T. Pederson, "Zero-Knowledge Authentication Scheme with Secret Key Exchange, "Advances in Cryptology-Crypto '88, Springer-Verlag, 1990, pp.583-588.
2. W. Diffie and M.E. Hellman, " New Directions in Cryptography," IEEE Transactions on Information Theory, Vol. IT-22, No. 6, Nov 1976, pp.644-654.
3. Kohnfelder, "Toward a Practical Public Key Cryptosystem, " Bachelor's thesis, MIT Department of Electrical Engineering, May 1978.
4. K. Koyama, "A Master Key for the RSA Public-Key Cryptosystem, " Transactions of the Institute of Electronics, Information, and Communication Engineers, Vol. J65D, No. 2, Feb. 1982, pp.163-170.
5. K. Koyama, "A Cryptosystem Using the Master Key for Multi-Address Communications, " Transactions of the Institute of Electronics, Information, and Communication Engineers, Vol. J65D, No. 9, Sep. 1982, pp.1151-1158.
6. M. Mambo, A. Nishikawa, S. Tsujii, and E. Okamoto, "Efficient Secure Broadcast Communication System, " Proceedings of the 1993 Korea-Japan Workshop on Information Security and Cryptography, Seoul, Korea, 24-26, Oct. 1993, pp. 23-33.
7. K. Ohta, "A Secure and Efficient Encrypted Broadcast Communication System Using a Public Master key, " Transactions of the Institute of Electronic, Information, and Communication Engineers, Vol. J70-D, No. 8, Aug. 1987, pp.1616-1624.
8. S.C. Pohlig and M.E. Hellman, "An Improved Algorithm for Computing Logarithms in GF(p) and Its Cryptographic Significance," IEEE Transactions on Information Theory, Vol. 24, No.1, Jan. 1978, pp.109-111.
9. R. L. Rivest, M.E. Hellman, J.C. Anderson, and J.W. Lyons, "A Method for Obtaining Digital Signatures and Public- Key

Cryptosystems, " Communications of the ACM, Vol. 21, No. 2, Feb. 1978, pp. 120-126.
10. R L. Rivest, A. Shamir, and L.M. Adleman, "On Digital Signatures and Public Key Cryptosystems, " MIT Laboratory for Computer Science, Technical Report, MIT/ LCS/ TR-212, Jan. 1979.
11. M. Steiner and G Tsudik, "New Multiparty Authentication Services and Key Agreement Protocols," Giuseppe Ateniese, IEEE Journal on Selected Areas in Communications, Vol. 18, No.4, April 2000, pp.628-639.
12. B. Schneier, "Applied Cryptography Second Edition," John Wiley & Sons, Inc. New York, U.S.A, 1996, pp.514.
13. B. Schneier, "Applied Cryptography Second Edition," John Wiley & Sons, Inc. New York, U.S.A, 1996, pp.515.