AUTHENTICATION PROTOCOL: METHODS REVIEW

Cahyo Crysdian and Abdul Hanan Abdullah
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia
Skudai 81310 Negeri Johor Darul Ta'zim, Malaysia
Emails: crysdian@lycos.com, hanan@fsksm.utm.my

Abstract: Authentication protocol as a part of security system has been growth rapidly since it was known that sending clear text password in the network is unsecured. Many protocols could be noted proposed to strengthen the authentication process. In 1985 an attempt to safeguard network services within Athena project resulting on the born of Kerberos [1][8], one of the protocol that has a lot of attention from the research community. Several years later researchers were discovered some weaknesses carried by this protocol [2][21]. In 1992, EKE was introduced by Bellovin and Merrit. Since that time, many protocols introduced could be considered as its variant [5][9][13][14]. Some other protocols such as OKE[15] and SRP[18] although claimed different from EKE, they have the same basic mechanism in holding authentication process. Here, we explain the mechanism of those protocols, their strength and their weaknesses and shortcomings. Due to the limitations of the number of paper pages, only two types of authentication protocol can be explained here i.e EKE and SRP.

1. Introduction

Authentication system has been receiving a wide attention from the researcher in the last decade. As a part of the security system, it becomes the basis of security mechanism since its output is used by the authorization mechanism to govern user in accessing system [23]. The main function of authentication is to identify someone who logs into the system. It ensures that the user logging in the system is really the one who he claims. Therefore, it becomes the door of the information system that determines whether the user is allowed to enter system or not.

In the past, authentication mechanism relies on the protection of clear text password that saves username and password in the plain text format in server. If a user log in to the system, he supplies a password and a username, then the system will compare those data with the data saved in the server. This mechanism has a lot of shortcoming in holding security. Someone who is logging into the server can get the password easily. Moreover, as the communication between client and server during the authentication does not encrypt the data being transmitted, an adversary eavesdropping the network traffic can also get the user password.

Although it has a lot of weaknesses, many information systems still keep using this method recently. And current condition shows that security factor becomes more critical, especially if the system is connected to the Internet. To cope with that condition, many stronger methods have been proposed to handle authentication. Some methods such as Kerberos [1][8], EKE [3] and its derivatives, OKE [15] and its derivatives and SRP [18], could be noted get a lot of attention from the research community, since they are believed able to solve the problem carried by the predecessor. This paper aims to discuss those authentication mechanisms, their weakness and drawback that is possible to resist the implementation of security system. The rest of this paper will be organized as follows. Section 2 and 3 discusses EKE and SRP respectively. The analysis on each authentication protocol is given in each section, whereas the conclusion of this paper is given in section 4.

2. EKE Family

EKE (Encrypted Key Exchange) originally was proposed by Bellovin and Merrit in 1992 to secure password against dictionary attack [3]. This protocol relies the combination of public key (asymmetric) and secret key (symmetric) cryptography to exchange secret information between two parties involved in the authentication. Later on [5] extended this protocol to become A-EKE (Augmented Encrypted Key Exchange) in order to deter the attack on the host where the password is stored by avoiding storing the password in clear text format. In 1996 [13] proposed SPEKE (Simple Password Encrypted Key Exchange) as the extension of EKE. It aims to build a strong authentication using only a small password. Besides preventing password from dictionary attack, this protocol proved able to achieve perfect forward secrecy, that is the disclosure of the password does not help the attacker to get the session key. Besides those protocols, there are also another variant of EKE such as M-EKE (Modified Encrypted Key Exchange) [9] and B-SPEKE [14]. A brief explanation of EKE is given as follows.

2.1. EKE (Encrypted Key Exchange)

[3] shows two way in implementing EKE, i.e. using public key and using exponential key exchange. Both will be explained briefly in this subsection. As stated earlier, this protocol uses a combination of public key and secret key cryptography in exchanging secret information between two parties involved in the authentication. To ease the explanation on its mechanism we use the following notation:

A,B	: Alice and Bob, the users
P	: The clear text password
E_A/D_A	: Public key pair

R	: Secret key
C_A	: Challenge generated by A
C_B	: Challenge generated by B
RA	: Random number generated by A
RB	: Random number generated by B
α	: A suitable Diffie-Hellman base
β	: A huge prime number for Diffie-Hellman
$H()$	: Hash function

a. EKE using public key

Before the authentication process started, Alice and Bob, two parties involve in the authentication share the knowledge of password P. Only those parties know the password. Using the notation denoted in this section above, authentication process between Alice and Bob can be described as the following:

i. $A \rightarrow B : A, \{E_A\}P$

Alice generates a pair random public key E_A/D_A, encrypt the public key E_A with the predefined password P, and sends the encryption to Bob. Alice also sends her name in the clear text format. Upon receiving the data, Bob decrypt it using P to obtain E_A.

ii. $B \rightarrow A : \{\{R\}E_A\}P$

Bob generates secret key R, encrypt it using E_A and password P, then sends the encryption to Alice. To obtain R, Alice decrypts the encryption using P and D_A.

iii. $A \rightarrow B : \{C_A\}R$

Alice generates a challenge C_A, and encrypts it with R then sends it to Bob.

iv. $B \rightarrow A : \{C_A, C_B\}R$

Bob decrypts C_A using R, generates a challenge C_B and encrypts it together with C_A using R and sends the encryption to Alice. Upon receiving the encryption, Alice decrypts it to obtain C_A and C_B. Then she verifies C_A to ensure that it has the same value with the one she generates previously.

v. $A \rightarrow B : \{C_B\}R$

Alice encrypts C_B using R and sends it to Bob. Bob decrypts the data using R to obtain C_B and verifies it in the same way Alice verifies C_A.

If all of those steps can be passed successfully, then the authentication is success. Otherwise it is failed. In this mechanism, the public key pair is E_A/D_A and the secret key is R, whereas the challenges (C_A and C_B) are used to prove that someone talking in the other side is really who he claims to be.

b. EKE using exponential key exchange

This protocol is known as Diffie-Hellman Encrypted Key Exchange (DH-EKE) that has the similar mechanism with EKE using public key. Rather than using R that is generated only by one side, here the key is generated by both sides as the result of exchanging process. The complete protocol can be explained as the follows:

i. $A \rightarrow B : A, \{\alpha^{RA}(mod\ \beta)\}P$

Alice generates a random number RA, calculates $\alpha^{RA}(mod\ \beta)$ and encrypts the result using P. Then sends the encryption to Bob together with her username in clear text format. Upon receiving the data, Bob decrypts it to obtain α^{RA}.

ii. $B \rightarrow A : \{\alpha^{RB}(mod\ \beta)\}P, \{C_B\}K$

Bob generates a random number RB, and then makes a computation to obtain K as the follows:

$$K = (\alpha^{RARB})(mod\ \beta) \quad (1)$$

Then Bob picks up a random challenge value C_B and encrypts that value with K. Bob also calculates $\alpha^{RB}(mod\ \beta)$ and encrypts the result using P. Both encryption results are sent to Alice. Upon receiving the message from Bob, Alice decrypts it and makes the same way as Bob to obtain K and C_B.

iii. $A \rightarrow B : \{C_A, C_B\}K$

Alice picks up a challenge value C_A, and encrypts it together with C_B using K. Then sends the result to Bob. Upon receiving the encryption Bob decrypts it and verifies the value of C_B. If C_B has the same value as he generates previously, it means he is talking to the right person.

iv. $B \rightarrow A : \{C_A\}K$

Bob encrypts C_A with K and sends it back to Alice. Upon receiving the data, Alice decrypts and verifies the result in the same way Bob verifies his challenge.

2.2. Discussion on EKE family

EKE could be noted introducing a unique way in safeguarding authentication process. There are two advantages delivered by this protocol, i.e. resists dictionary attack and achieves perfect forward secrecy. However, EKE and its variants still carry some drawbacks and weaknesses as reported by researchers as follow. [9] notes that original EKE is susceptible to Denning-Sacco attack although it could be fixed in M-EKE. [17] shows that EKE is easy to get denial of service. [18] stresses that the password shared between two parties in the clear-text format is the greatest failing of EKE, since it can lead to the possibility that the password is stolen. Whereas the refinement of holding clear-text password such as done to B-SPEKE [14] by adding another key exchange just bring a more complex computational and increase running time. Moreover, it is also worth to note that EKE is not comfortable for the purpose of frequent and lightweight authentication. To complete authentication process, both parties engage in intensive communication to exchange information. Therefore, a dependency on the network is great. This method cannot be employed in a unreliable networking environment such as accessing server through internet since it leads to low bandwidth and communication distortion.

3. SRP (Secure Remote Password)

SRP protocol was introduced by Thomas Wu in 1998. This protocol is constructed based on AKE (Asymmetric Key Exchange) [18]. In term of its function, AKE has the similarity with EKE in the exchanging keys between two parties and using that key to verify each other about the knowledge of the password. But it avoids the usage of encryption in the exchanging process. SRP as the AKE construction follows the function of AKE. The explanation below will directly describe about SRP, since it initially covers the explanation of AKE. The notations used in explaining this protocol are:

n	: a large prime number
g	: a primitive root modulo n (a generator)
s	: user's salt
P	: user's password
x	: private key derived from the password and salt
v	: user's verifier
u	: random parameter generated by host
a,b	: random ephemeral private key
A,B	: public key gotten from a and b
$H()$	: one way hash function
K	: session key
Carol $\rightarrow$ *Steve* : A	: Carol sends Steve the value of A

Let us assume that there are two parties, Carol and Steve in the authentication process. Assumed that Carol is the client who needs to get the service, whereas Steve is the host that authenticates client. Carol has a password P in which she never reveals to any one. To share that information to Steve, she computes:

$$x = H(s,P) \qquad (2)$$

$$v = g^{x} \qquad (3)$$

and let Steve stores the value of v and s (her verifier and salt). Then if Steve needs to authenticate Carol, they take the following steps:

i. *Carol* $\rightarrow$ *Steve : carol*

Carol reveals her user name to Steve.

ii. *Steve* $\rightarrow$ *Carol* : s

Steve sends her salt to Carol. Upon receiving s, Carol computes x using equation (2).

iii. *Carol* $\rightarrow$ *Steve* : A

Carol generates a random number a and computes:

$$A = g^{a} \qquad (4)$$

Then sends A to Steve.

iv. *Steve* $\rightarrow$ *Carol* : B,u

Steve generates random number b and u and computes:

$$B = v + g^{b} \qquad (5)$$

Then sends B to Carol together with u.

v. Based on the value held, each side computes as follow:

For Carol : $S = (B - g^{x})^{(a+ux)}$ (6)

For Steve : $S = (A \cdot v^{u})^{b}$ (7)

Both side will get the same value of S if the password used in step 2 is the same with the password used in computing Carol's verifier stored by Steve. Then both side get the session key K by putting S in the hash function as follow:

$$K = H(S) \qquad (8)$$

vi. *Carol* $\rightarrow$ *Steve* : $M[1]$

Carol computes $M[1]$ by inputting A, B and K in the hash function:

$$M[1] = H(A,B,K) \qquad (9)$$

and sends the value of $M[1]$ to Steve for verification. Upon receiving that value Steve verifies $M[1]$.

vii. *Steve* $\rightarrow$ *Carol* : $M[2]$

Steve computes $M[2]$ by inputting A, $M[1]$ and K in the hash function:

$$M[2] = H(A,M[1],K) \qquad (10)$$

and sends the value of $M[2]$ to Carol for verification. Upon receiving that value Carol verifies $M[2]$. If $M[2]$ gotten by Carol is matched with the value she gets from Steve, then the authentication is success.

3.1. Discussion on SRP

The intention of this protocol is to avoid the encryption in exchanging information to gain some other advantages that could be fined in [18]. Although it could be constructed in an elegant way using mathematical equation, avoiding encryption opens the hole in the security mechanism, as eavesdropper can get some information running on the network. It lets the attacker to get more worth information by supplying that information to its mechanism. Using this way an adversary is not just able to get the session key, but it is also possible to get the password although the password has never been revealed to other party.

Here is the way in which the attack can be committed. Let us take a look in the exchanging process. The information transmitted between both sides is not encrypted. An eavesdropper can easily get the value of salt, A, B and u. Furthermore, he also can catch $M[1]$ and $M[2]$. Those values can be used to commit a serious attack. An adversary can computes S using the following formula:

$$S = g^{(ab+bux)} \qquad (11)$$

Referring to equation (5) and based on the value he holds, he can get the following equation to compute S.

$$S = (B - v)^{(a+ux)} \qquad (12)$$

From equation (12) the adversary still has not gotten several parameter, i.e. a, v and x. But it does not prevent the attack. He can get the value of a using looping computation[1] on equation (4). It is done by supplying each value of a in equation (4) and compare the result to the value of A he gets from the network. If both value is matched, it means he get the same value of ephemeral random value a as generated by Carol. Whereas user's verifier (v) and x come from one source, i.e. user's password (P). Thus by knowing x he easily can get the value of user verifier using equation (3). But how to get x? Basically, it takes the same way to get the value of a, that is using looping computation. Instead of just computing one equation, an adversary needs to compute equation (2), (3), (12), (8) and (9) consecutively. Then compare the result to the $M[1]$ he catches from the network. To make it clear

[2] *Looping computation is the attack committed in the similar way as dictionary attack. Instead of just using the collection of words and making a direct guess to breach the password, it supplies both words and numbers to the predefined equations and matches the result to the reference value.*

about this attack, we re-list the equation in which the adversary must computes as follows:

i. $x = H(s,P)$

ii. $v = g^x$

iii. $S = (B - v)^{(a+ux)}$

iv. $K = H(S)$

v. $M[1] = H(A,B,K)$

So, he can commits a dictionary attack by supplying each candidate of *P*, follows those equation and finally compares the result to *M[1]*. If the result is matched to *M[1]*, it means he has gotten not only the password, but also the session key.

4. Conclusion

It is common happened in the real live that someone reveal his password to somebody else, regardless it is accidentally or intentionally. Many reasons can be given for that condition, such as conducting user support, holding system maintenance, etc. Whatever the reason, this condition must not be allowed by the system. Authentication mechanism should be able to deter the usage of someone else password to get access from the system.

The authentication protocols explained in section 2 and 3 do not have any protection from this violation. In those protocols, having a correct password means getting the access to the system. There is no further protection to ensure that somebody logging in the system is the authorized user. Therefore it is necessary to build the authentication system that applies "Knowing somebody else password will not guarantee having an access to the system".

5. References

[1] Jennifer G. Steiner, Clifford Neuman, Jeffrey I. Schiller. Kerberos: An Authentication Service for Open Network Systems. *Usenix Conference Proceedings*. 1988.

[2] S.M. Bellovin and M. Merit. Limitations of the Kerberos Authentication System. *In Proceedings of the 1991* Winter *USENIX Conference*. 1991.

[3] S.M. Bellovin and M. Merit. Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks. *In Proceedings of the 1992 IEEE Computer Society Conferenc e on Research in Security and Privacy*. 1992.

[4] Li Gong, T. Mark A.. Lomas, Roger M. Needham and Jerome H. *Saltzer*. Protecting Poorly Chosen Secrets from Guessing Attacks. *In IEEE Journal on Selected Areas in Communications, Vol. 11, Number 5*. June 1993.

[5] S.M. Bellovin and M. Merit. Augmented Encrypted Key Exchange: A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise. *Technical Report, AT&T Bell Laboratories*. 1994.

[6] John T. Kohl, B. *Clifford* Neuman and Theodore Ts'o. The evolution of the Kerberos Authentication Service. *In Distributed Open Systems, IEEE Computer Society Press*. 1994.

[7] Thomas Y. C. Woo and Simon S. Lam. A Lesson on Authentication Protocol Design. *In ACM Operating Systems Review, Vol.28, Number 3*. July 1994.

[8] B. Clifford Neuman and Theodore Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications Magazine, Volume 32, Number 9*. September 1994.

[9] M. Steiner, G. Tsudik and M. Waidner. Refinement and Extension of Encrypted Key Exchange. *Operating Systems Review*. 1995.

[10] A. Menezes, P. van Oorschot and S. Vanstone. Handbook of Applied Cryptography. *CRC Press*. 1996.

[11] Gavin Lowe. Some New Attacks upon Security Protocols. *In Proceedings of the 9th IEEE* Computer *Security Foundations Workshop, pages 162-169*. June 1996.

[12] J A Clark and J L Jacob. Attacking Authentication Protocols. *High Integrity Systems 1(5):465-474*. August 1996.

[13] David Jablon. Strong Password-Only Authenticated Key Exchange. *Computer Communication Review*. 1996.

[14] David Jablon. Extended Password Key Exchange Protocols Immune to Dictionary Attacks. *In* WETICE'97 *Enterprise Security Workshop*. 1997.

[15] Stefan Lucks. Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys. *In Proceedings of the Workshop on Security Protocols*. 1997.

[16] Gavin Lowe. A Family of Attacks upon Authentication Protocols. *Technical Report 1997/5,* Department *of Mathematics and Computer Science, University of Leicester*, 1997.

[17] Sarvar Patel. Number Theoretic Attacks on Secure Password Schemes. 1997 IEEE Symphosium on Security and Privacy. 1997

[18] Thomas Wu. The Secure Password Protocol. *In Proceedings of the Internet Security Network* and *Distributed System Security Symposium*. March 1998.

[19] Radia Perlman and Charlie Kaufman. Secure *Password-*Based Protocol for Downloading a Private Key. *Proceedings of the 1999 Network and Distributed System Security Symposium*. 1999.

[20] Seungjoo Kim, Byungchun Kim and Sungjun Park. Comments on *Password*-Based Private Key Download of NDSS'99. *Electronics Letters 35(22), IEE Press*. 1999

[21] Thomas Wu. A Real-World Analysis of Kerberos Password Security. *Proceedings of the 1999 Network and Distributed System Security*. February 3-5, 1999.

[22] P. MacKenzie and R. Swaminathan. Secure Network Authentication with Password Identification. *Presented to IEEE P1363a*. August, 1999

[23] Silvana Castano, Maria Grazia Fugini, Giancarlo Martella, Pierangela Samarati. Database Security. *Addison Wesley*. 1994.