Available online at www.sciencedirect.com
ScienceDirect
Journal of Computer and System Sciences 72 (2006) 1002–1011
JOURNAL OF COMPUTER AND SYSTEM SCIENCES
www.elsevier.com/locate/jcss

Password-based authentication and key distribution protocols with perfect forward secrecy

Hung-Min Sun [a,*], Her-Tyan Yeh [b]

[a] *Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan*
[b] *Department of Computer and Communication, Southern Taiwan University of Technology, Tainan 710, Taiwan*

Received 28 March 2004; received in revised form 23 January 2006

Available online 15 May 2006

Abstract

In an open networking environment, a workstation usually needs to identify its legal users for providing its services. Kerberos provides an efficient approach whereby a trusted third-party authentication server is used to verify users' identities. However, Kerberos enforces the user to use strong cryptographic secret for user authentication, and hence is insecure from password guessing attacks if the user uses a weak password for convenience. In this paper, we focus on such an environment in which the users can use easy-to-remember passwords. In addition to password guessing attacks, perfect forward secrecy (PFS in short) is another important security consideration when designing an authentication and key distribution protocol. Based on the capability of protecting the client's password, the application server's secret key, and the authentication server's private key, we define seven classes of perfect forward secrecy and focus on protocols achieving class-1, class-3, and class-7 due to their hierarchical relations. Then, we propose three secure authentication and key distribution protocols to provide perfect forward secrecy of these three classes. All these protocols are efficient in protecting poorly-chosen passwords chosen by users from guessing attacks and replay attacks.

Keywords: Network security; Authentication; Perfect forward secrecy; Guessing attack; Password

1. Introduction

In today's distributed computing environment, secure communication in insecure communication channels is a very important issue. For this reason, user authentication and secret key distribution become the most important security service for communication networks. Therefore, authentication and key distribution protocols are necessary in distributed environments.

Password-based mechanism has been the most widely used method for user authentication since it allows people to choose and remember their own passwords without any assistant device. However, human users usually choose easy-to-remember passwords so that they are vulnerable to password guessing attacks. On the contrary, the entities excluding human users, such as servers, can directly use strong cryptographic secrets for entity authentication and hence prevent password guessing attacks.

* Corresponding author. Fax: +886 3 5723694.
E-mail address: hmsun@cs.nthu.edu.tw (H.-M. Sun).

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2006.03.005

In recent years, a variety of protocols for authentication and key distribution have been proposed and applied to many communication systems. Diffie and Hellman [18] described how to establish a common session key by public messages. Needham and Schroeder [19] proposed a point protocol that uses encryption to achieve authentication. In 1992, Bellovin and Merritt [1] presented a new protocol known as Encrypted Key Exchange, or EKE in short. It is the landmark of 2-party authentication and key exchange protocols [1–7]. EKE can resist guessing attacks by giving the attacker insufficient information to verify a guessed password. EKE performs a key exchange as well, so both parties can encrypt their transmissions once authentication is established.

On the other hand, Gong, Lomas, Needham, and Saltzer [8] proposed a protocol, called GLNS protocol, in a three-party setting in which two users (clients) establish a session key through an authentication server. Timestamps are used in the protocol to guarantee message freshness. By using nonces and confounders, the protocol is successful in generating a large search space to resist off-line password guessing attacks. Later, Gong proposed an optimal version [9] of the GLNS protocol, which reduces the amount of message transmissions and does not require timestamps. Keung and Siu [10] proposed another protocol that is immune to replay attacks and off-line password guessing attacks. Kwon, Kang and Song [11] also proposed another protocol for mutual authentication and key distribution. In these two protocols, the concept of one-time pad and one-way hash function is applied to reduce the computation cost.

Up to now, most of the literature on three-party authentication and key distribution protocols [7–16] have focused on the environment in which two users (clients) establish a session key through an authentication server (*client–client–server model*). However, let us consider an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. In this setting, we have a centralized server (authentication server) for granting permission to user (client) to access some services in other servers (application server). This model consists of two servers with a client. To distinguish this with the existing model, we call this a *client–server–server model*. In this environment, a workstation cannot be trusted to identify its users correctly to network services. In particular, the following three threats exist:

- A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
- A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
- A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access. Rather than building in elaborate authentication protocols at each server, a centralized authentication server is provided to authenticate users to servers and servers to users. Such an environment is suitable for many applications. A typical example used in such a setting is the well-known protocol, Kerberos [17]. However, Kerberos enforces the user to use a strong cryptographic secret for user authentication, and hence is insecure from password guessing attacks if the user uses a weak password. In this paper, we focus on such an environment in which the users use easy-to-remember passwords. In addition to password guessing attacks, perfect forward secrecy (PFS in short) is another important security consideration when designing an authentication and key distribution protocol. Based on the capability of protecting client's password, the application server's secret key, and the authentication server's private key, we define seven classes of perfect forward secrecy. Among them, we are interested in class-1, class-3, and class-7 PFS because they dominate the security from low level to high level. We also propose three protocols that can fit this environment, resist various attacks, and separately achieve these three classes of PFS.

The remainder of this paper is organized as follows. Section 2 summarizes the description of notations and security requirements. In Section 3, a protocol for class-1 PFS is proposed. We present a protocol for class-3 PFS in Section 4. In Section 5, a protocol for class-7 PFS is suggested. In Section 6, we compare these three newly proposed protocols with some well-known protocols. Finally we conclude this paper in Section 7.

2. Notations and security requirements

2.1. Notations

The notations in Table 1 are used throughout this paper.

Table 1
Notations

A	Client (user)
B	Application server
S	Authentication server
P_A	Password shared between A and S
S_B	Secret key shared between B and S
K_S	Public key of the authentication server
x, y, ra, rb, a, b, rb'	Random numbers
$h(\,)$	One-way hash function
$A \rightarrow B : M$	A sends a message M to B
g	Base generator
P	Large prime (P is the modulus of all modular exponentiations)
$[\text{info}]_K$	Symmetric encryption of "info" with key K
$\{\text{info}\}_K$	Asymmetric encryption of "info" with public key K

2.2. Security requirements

In this paper, we consider some well-known attacks including password guessing attacks and replay attacks, and define seven classes of perfect forward secrecy.

- Password guessing attacks:
  Password guessing attacks can be classified into two types:
  (1) On-line password guessing attacks: An attacker tries to use a guessed password to pass the verification of the authentication server in an on-line manner. Generally, the authentication server can detect such an attack by noticing continuous authentication failures.
  (2) Off-line password guessing attacks: An attacker eavesdrops communication messages during a protocol and stores them locally. Then he/she tries to find out the weak password by repeatedly guessing a possible password and verifying the correctness of the guess via the captured information in an off-line manner. In general, such an attack can be prevented only by carefully designing the protocol such that no verifiable information can be used by the attack to verify the correctness of one guess on password.
- Replay attacks:
  In this attack, an adversary tries to replay messages partially or completely obtained in previous communications. If he can impersonate other users or expose other secret that is sensitive and useful for further deceptions, by guessing attacks, known-plaintext attacks or other cryptographic analysis methods, then the protocol is said to be vulnerable to replay attacks.
- Perfect forward secrecy (PFS):
  In a two-party setting, a password-based protocol is called perfect forward secure [5,8,16] if the revealing of the password to an attacker does not help him obtain the session keys of past sessions. Here we classify perfect forward secrecy in a three-party setting according to the cases of the combinations whether the client's password, the application server's secret key, and the authentication server's private key are revealed. Based on the capability of protecting the client's password, the application server's secret key, and the authentication server's private key, we define seven classes of perfect forward secrecy. In Table 2, we show these seven classes of perfect forward secrecy. For example, a protocol providing class-1 PFS means that an adversary cannot extract the past session keys from previous communications despite the reveal of real password, as long as the application server's secret key and the authentication server's private key are still kept secret.

Note that the risk of revealing the client's password is much higher than that of revealing the application server's secret key, and the risk of revealing the application server's secret key is much higher than that of revealing the authentication server's private key. Due to their hierarchical relations among class-1 PFS, class-3 PFS, and class-7 PFS, we are interested in designing protocols satisfying these three classes. The detailed definitions for class-1 PFS, class-3 PFS, and class-7 PFS are in the following.

Table 2
Seven classes of perfect forward secrecy

	Client's password	Application server's secret key	Authentication server's private key
Class-1 PFS	**Revealed**	**Secure**	**Secure**
Class-2 PFS	Secure	Revealed	Secure
Class-3 PFS	**Revealed**	**Revealed**	**Secure**
Class-4 PFS	Secure	Secure	Revealed
Class-5 PFS	Revealed	Secure	Revealed
Class-6 PFS	Secure	Revealed	Revealed
Class-7 PFS	**Revealed**	**Revealed**	**Revealed**

- Class-1 PFS (PFS with low security):
  A protocol providing class-1 PFS means that if the client's password is revealed to an attacker, but the application server's secret key and the authentication server's private key are still secure, it does not help the attacker obtain the session keys of previous sessions.
- Class-3 PFS (PFS with medium security):
  A protocol providing class-3 PFS means that if the client's password and the application server's secret key are simultaneously revealed to an attacker, but the authentication server's private key is still secure, it does not help the attacker obtain the session keys of previous sessions.
- Class-7 PFS (PFS with high security):
  A protocol providing class-7 PFS means that if the client's password, the application server's secret key, and the authentication server's private key are simultaneously revealed to an attacker, it still does not help the attacker obtain the session keys of previous sessions.

3. A protocol providing PFS with low security

An efficient authentication and key distribution protocol providing class-1 PFS is proposed in this section. There are three principals involved in our protocol: an application server B who provides services to clients, a client A who requests services from the application server, and an authentication server S who is responsible for authentication and who distributes the common session key shared between the client A and the application server B.

3.1. The proposed protocol

In this protocol, we assume that all principals know the server's public key K_S in the system. We also assume that a poorly chosen password P_A chosen by A is known to S via a secure channel. Similarly, the application server's secret key S_B is known to S via a secure channel.

We show our protocol in Fig. 1 and the detailed steps are described as follows:

(1) $A \to S$: $A, \{A, B, P_A, ra\}_{K_S}$:
A chooses a random number ra and keeps it secret. Then, A encrypts A, B, P_A, ra with server S's public key K_S and transmits the encrypted message as a request to S, where P_A is the password of A.

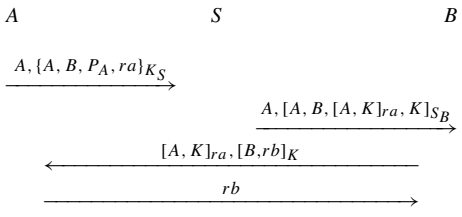

Fig. 1. A protocol providing PFS with low security.

(2) $S \to B$: $A, [A, B, [A, K]_{ra}, K]_{S_B}$:
After receiving client A's message, the authentication server, S, decrypts $\{A, B, P_A, ra\}_{K_S}$ with his private key corresponding to the public key K_S and checks the authenticity of A by verifying A's password P_A. Then, he chooses a common key K. Hence, he can compute $[A, B, [A, K]_{ra}, K]_{S_B}$, and transmit it to B. Note that the value ra also acts as a one-time key.

(3) $B \to A$: $[A, K]_{ra}, [B, rb]_K$:
The application server B first decrypts the message $[A, B, [A, K]_{ra}, K]_{S_B}$ with his secret key S_B and gets the common key K. Then B chooses a challenge value rb, encrypts B and rb with the common key K, and sends $[A, K]_{ra}$ and $[B, rb]_K$ to the user A.

(4) $A \to B$: rb:
In step 4, the user A decrypts message $[A, K]_{ra}$ with ra and gets the common key K. Then, he decrypts $[B, rb]_K$, checks the validity of K, and sends the response value rb to B.
Finally, the user A and the application server B can authenticate each other and compute the common session key $h(K)$.

3.2. Security analysis

3.2.1. Guessing attacks

In this protocol, we protect the password from an attacker, who can be either malicious or merely incompetent. If the attacker attempts to use a guessed password in an on-line transaction, a failed guess can be detected and logged. So, our scheme is resistant to on-line guessing attack.

Considering an off-line guessing attack, A's password is used only to authenticate A's status in Message 1. It is not included in any verifiable data. Thus, an attacker is impossible to verify his guess on the password unless he can know the random number ra. Therefore, our protocol is immune to off-line password guessing attacks.

3.2.2. Replay attacks

Although an attacker can replay an old Message 1 because the server cannot decide its freshness, all the attacker can get is $\{[A, B, [A, K]_{ra}, K]_{S_B}, [A, K]_{ra}, [B, rb]_K\}$. Because he is unable to know the random numbers ra included in Message 1 or server B's secret key S_B to decrypt these messages, this does not help him compromise a future session key K' or to guess the password. Thus, our protocol is secure against message replay attacks.

3.2.3. Class-1 PFS

Here, we consider whether the proposed scheme provides class-1 PFS. When a user's password is revealed, an attacker can know P_A. Because the attacker does not know the server S's private key, he cannot decrypt Message 1 to get ra. Also, he cannot decrypt Message 2 because he does not know the server B's secret key. So the attacker does not have any opportunity to obtain K and get the session key $h(K)$. Therefore, the session key is still secure. Therefore, our scheme provides class-1 PFS.

4. Protocol providing PFS with medium security

In this section, we propose an efficient authentication and key distribution protocol providing class-3 PFS.

4.1. The proposed protocol

We show our protocol in Fig. 2 and the detailed steps are described as follows.

(1) $A \to B$: $A, \{A, B, P_A, ra\}_{K_S}$:
A chooses a random number ra and keeps it secret. Then, A encrypts A, B, P_A, and ra with the server S's public key K_S and transmits the encrypted message as a request to the server B, where P_A is the password of A.

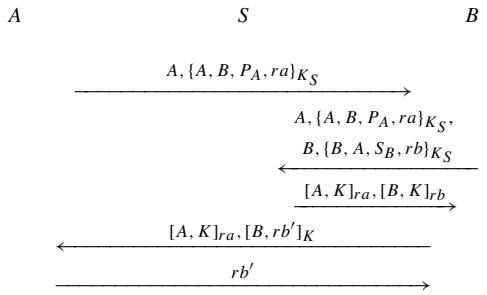

Fig. 2. A protocol providing PFS with middle security.

(2) $B \to S$: $A, \{A, B, P_A, ra\}_{K_S}, B, \{B, A, S_B, rb\}_{K_S}$:
After receiving the client A's message, the server B chooses a confounder rb, and encrypts B, A, S_B, and rb with the server S's public key K_S. Both ciphertexts $\{A, B, P_A, ra\}_{K_S}$ and $\{B, A, S_B, rb\}_{K_S}$ together with A and B are sent to the server S.

(3) $S \to B$: $[A, K]_{ra}, [B, K]_{rb}$:
After receiving Message 2, the authentication server S decrypts it with his private key, then checks the authenticity of A by verifying A's password P_A and the authenticity of B by verifying B's secret key S_B. The server S then chooses a common key K, computes $\{[A, K]_{ra}, [B, K]_{rb}\}$, and transmits it to B. Note that the values ra and rb also act as one-time keys.

(4) $B \to A$: $[A, K]_{ra}, [B, rb']_K$:
The application server B first decrypts the message $[B, K]_{rb}$ with rb and gets the common key K. Then B uses rb' as a challenge value, encrypts B, rb' with the common key K and sends $[A, K]_{ra}$ and $[B, rb']_K$ to the client A.

(5) $A \to B$: rb':
In step 5, the client A decrypts message $[A, K]_{ra}$ with ra and gets the common key K. Then, he decrypts $[B, rb']_K$, checks the validity of K, and sends the response value rb' to B.
After authentication procedure, the client A and the application server B negotiate a session key $h(K)$ to communicate with each other securely.

4.2. Security analysis

4.2.1. Guessing attacks

In this protocol, the password guessing attacks cannot succeed because no poorly chosen password is used as encryption key. So, our protocol is immune to password guessing attacks.

4.2.2. Replay attacks

Although the attacker can replay old Message 1 and Message 2, this does not help him compromise a future session key K' from S's reply because ra and rb are unknown to the attacker. Thus, our protocol is secure against message replay attacks.

4.2.3. Class-3 PFS

Class-3 PFS is discussed in this section. We assume that the client A's password P_A and the server B's secret key S_B are known by an attacker. Because the attacker does not know the server S's secret key to decrypt Message 1 or Message 2 in order to get ra or rb, the attacker does not have any opportunity to obtain K and get the session key $h(K)$. The session key is still secure. Therefore, our scheme provides class-3 PFS.

5. Protocol providing PFS with high security

5.1. The proposed protocol

In Fig. 3, the third authentication and key distribution protocol is provided to gratify Class-7 PFS.

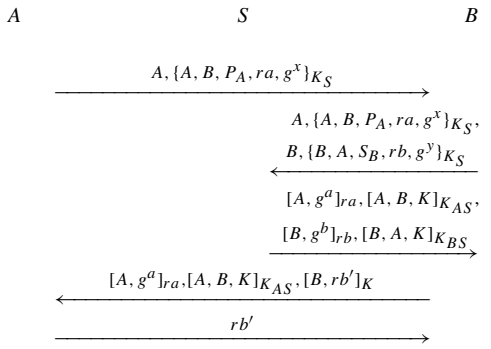

Fig. 3. A protocol providing PFS with high security.

The details are described as follows:

(1) $A \to B$: $A, \{A, B, P_A, ra, g^x\}_{K_S}$:
A chooses a confounder ra, a random number x and computes g^x. Then A encrypts A, B, P_A, ra, g^x by the server S's public key K_S and sends the ciphertext to B, where P_A is the password of A.

(2) $B \to S$: $A, \{A, B, P_A, ra, g^x\}_{K_S}, B, \{B, A, S_B, rb, g^y\}_{K_S}$:
After receiving the client A's message, the server B chooses a confounder rb, and computes g^y mod P by choosing a random number y. Then, he encrypts B, A, S_B, rb, g^y with the server S's public key K_S. Both ciphertexts $\{A, B, P_A, ra, g^x\}_{K_S}$ and $\{B, A, S_B, rb, g^y\}_{K_S}$ together with A and B are sent to the server S.

(3) $S \to B$: $[A, g^a]_{ra}, [A, B, K]_{K_{AS}}, [B, g^b]_{rb}, [B, A, K]_{K_{BS}}$:
After receiving Message 2, the authentication server S decrypts it with his private key. S checks the authenticity of A by verifying A's password P_A and the authenticity of B by verifying B's secret key S_B. He then chooses a common key K, computes $\{[A, g^a]_{ra}, [A, B, K]_{K_{AS}}, [B, g^b]_{rb}, [B, A, K]_{K_{BS}}\}$, and transmits it to B, where a and b are chosen by S, $K_{AS} = (g^x)^a = (g^a)^x = g^{xa}$ and $K_{BS} = (g^y)^b = (g^b)^y = g^{yb}$ are used to securely pass the session key K. Note that the values ra and rb also act as one-time keys.

(4) $B \to A$: $[A, g^a]_{ra}, [A, B, K]_{K_{AS}}, [B, rb']_K$:
The application server B first decrypts the message $[B, g^b]_{rb}$ with rb and computes $K_{BS} = (g^b)^y = g^{yb}$. He then decrypts $[B, A, K]_{K_{BS}}$ with K_{BS} and gets the common key K. Then, B uses rb' as a challenge value, encrypts B, rb' with the common key K and sends $[A, g^a]_{ra}, [A, B, K]_{K_{AS}}, [B, rb']_K$ to the client A.

(5) $A \to B$: rb':
The client A decrypts the message $[A, g^a]_{ra}$ with ra and computes $K_{AS} = (g^a)^x = g^{xa}$. Then, he decrypts $[A, B, K]_{K_{AS}}$ with K_{AS} and gets the common key K. After that, he decrypts $[B, rb']_K$, checks the validity of K, and sends the response value rb' to B.
Finally, the client A and the application server B can authenticate each other and compute the common session key $h(K)$.

5.2. Security analysis

5.2.1. Guessing attacks

Instead of authenticating the status of A, the client's password is not used in this protocol. This leads to that an attacker has not any verifiable data to make sure whether his guess is right or wrong. So, our protocol is immune to password guessing attacks.

5.2.2. Replay attacks

An attacker can replay old Message 1 and Message 2 because the server cannot decide its freshness. However, no additional information can help him compromise a future session key or guess the password from S's reply. Thus, our protocol is secure against message replay attacks.

5.2.3. Class-7 PFS

Our protocol is based on the following well-known hard problem, which is believed infeasible to solve in polynomial time.

Diffie–Hellman problem [18]: Given a prime P, a generator g, and two numbers $g^x \bmod P$ and $g^y \bmod P$, find $g^{xy} \bmod P$.

We assume that the client A's password P_A, the application server B's secret key S_B, and the authentication server S's private key are all known by an attacker. Then the attacker can decrypt Message 2 to obtain ra, g^x, rb, g^y, and use ra and rb to decrypt part of Message 3 to obtain g^a, g^b. But he cannot calculate K_{AS} or K_{BS} because the difficulty is similar to solve the Diffie–Hellman problem [18]. So the attacker does not have any opportunity to obtain K and get the session key $h(K)$. Thus the session key is still secure. Therefore, our scheme provides class-7 PFS.

6. Efficiency and comparison

In this paper we focus on an environment with three parties (client–server–server model) that the user and the application server authenticate each other via the authentication server. Because there is not any existing protocol designed for this environment, one may employ some traditional three-party (client–client–server model) protocols for this environment by putting one of the clients as the application server. Therefore, we compare some well-known traditional three-party protocols including the optimal GLNS protocol [9], the improved K1P protocol [13], and the extension of Otway–Rees protocol [12], with our three newly proposed protocols (low PFS protocol, medium PFS protocol and high PFS protocol). We focus on several items such as the shared secret types, the number of steps, the number of random numbers, the number of symmetric encryption operations, and the number of asymmetric encryption operations. We ignore other comparisons like the total amount of data transferred because these items are varying for different security levels. Table 3 shows the comparison results.

The extension of Otway–Rees protocol does not belong to any class of PFS. Compared with the optimal GLNS protocol and the improved K1P protocol, our medium PFS protocol (class-3 PFS) has the least number of random numbers, the least number of symmetric encryption operations, and the same number of steps and asymmetric encryption operations. The results show that our medium PFS protocol is the most efficient protocol among these class-3 PFS protocols. In addition, our high PFS protocol can provide class-7 PFS while the others cannot.

We remark that by applying an existing three-party client–client–server protocol with class-3 PFS, e.g. [9] and [13], to the client–server–server model with class-3 PFS, one may treat one of the clients as the application server. Instead of using a password, a strong key (secret key) can be used in the application server. This seems to imply that we do not need to design new protocols for the client–server–server model. However, there are three points for us to promote new protocols for the client–server–server model. First, a client–client–server protocol is designed to cope with password guessing attacks in both clients. Now, if one of the clients, that is the application server, can use a strong key as its secret, one side of password guessing attacks can be always avoided. Thus we believe that the cost of a client–server–server protocol can be less than that of a client–client–server protocol. This can be seen by comparing our medium PFS protocol with the optimal GLNS protocol [9] and the improved K1P protocol [13] in Table 3. Secondly, if one simply wants to use a client–server–server protocol with class-1 PFS, our low PFS protocol can achieve. Although our medium PFS protocol, the optimal GLNS protocol [9], and the improved K1P protocol [13] can also achieve class-1 PFS (this is because achieving class-3 PFS implies achieving class-1 PFS), these protocols are not designed specifically for class-1 PFS and hence incur cost-inefficiency. Thirdly, most of the existing client–client–server protocols achieve class-3 PFS. Therefore if we need a client–server–server protocol providing class-7 PFS, we cannot apply those client–client–server protocols to the client–server–server model. Instead, we need design a new protocol which meets the required security level. To sum up, as reflected from the comparison table, the tailor-made protocols proposed in this paper have significant advantages over converting existing client–client–server protocols.

7. Conclusions

An open distributed environment in which the clients access services on application servers through an authentication server is discussed in this paper. Perfect forward secrecy is one of the most important considerations of designing an authentication and key distribution protocol to fit this environment. Based on the capability of protecting the client's password, the application server's secret key, and the authentication server's private key, we define seven classes of

Table 3
Comparison of the well-known protocol

	C1	C2	C3	C4	C5	C6
Optimal GLNS [9]	Class 3 PFS	A: Password B: Password S: Private key	5	10	A: 2 B: 2 S: 2	A: 1 B: 1 S: 0
Improved K1P [13]	Class 3 PFS	A: Password B: Password S: Private key	5	5	A: 1 B: 1 S: 2	A: 1 B: 1 S: 0
Extension of Otway–Rees [12]	×	A: Secret key B: Secret key S: Secret key	5	3	A: 2 B: 2 S: 2	A: 0 B: 0 S: 0
Low PFS protocol	Class 1 PFS	A: Password B: Secret key S: Private key	4	2	A: 0 B: 1 S: 2	A: 1 B: 0 S: 0
Medium PFS protocol	Class 3 PFS	A: Password B: Secret key S: Private key	5	3	A: 0 B: 1 S: 2	A: 1 B: 1 S: 0
High PFS protocol	Class 7 PFS	A: Password B: Secret key S: Private key	5	7	A: 0 B: 1 S: 2	A: 1 B: 1 S: 0

Notes: C1: PFS; C2: Secret used for authentication; C3: Steps; C4: Random numbers; C5: Symmetric encryption; C6: Asymmetric encryption.

perfect forward secrecy. We focus only on class-1 PFS, class-3 PFS, and class-7 PFS due to their hierarchical relations, and propose three authentication and key distribution protocols to provide them, respectively. Of course, they all also resist various attacks such as password guessing attacks and replay attacks.

Acknowledgments

This research was supported in part by the National Science Council, Taiwan, under contract NSC94-2213-E-007-039. We are grateful to anonymous reviewers for their valuable comments.

References

[1] S. Bellovin, M. Merritt, Encrypted key exchange: Password-based protocols secure against dictionary attacks, in: Proc. of IEEE Symposium on Research in Security and Privacy, Oakland, May 1992.
[2] S. Bellovin, M. Merritt, Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise, AT&T Bell Laboratories, 1993.
[3] D. Jablon, Strong password-only authentication key exchange, Comput. Commun. Rev. 26 (5) (October 1996) 5–26.
[4] D. Jablon, Extended password key exchange protocols immune to dictionary attack, in: Proc. of the WETICE Workshop on Enterprise Security, Cambridge, MA, June 1997.
[5] S. Lucks, Open key exchange: How to defeat dictionary attacks without encrypting public keys, in: Proc. of the Security Protocol Workshop, Springer-Verlag, April 1997.
[6] T. Wu, The secure remote password protocol, in: Internet Society Symposium on Network and Distributed System Security, 1998.
[7] T. Kwon, J. Song, Efficient key exchange and authentication protocol protecting weak secrets, IEICE Trans. Fundamentals E81-A (1) (January 1998) 156–163.
[8] L. Gong, M. Lomas, R. Needham, J. Saltzer, Protecting poorly chosen secrets from guessing attacks, IEEE J. Sel. Areas Comm. 11 (5) (1993) 648–656.
[9] L. Gong, Optimal authentication protocols resistant to password guessing attacks, in: Proc. of the 8th IEEE Computer Security Foundation Workshop, County Kerry, Ireland, June 1995.
[10] S. Keung, K. Siu, Efficient protocols secure against guessing and replay attacks, in: Proc. of the Fourth International Conference on Computer Communications and Networks, 1995, pp. 105–112.
[11] T. Kwon, M. Kang, J. Song, An adaptable and reliable authentication protocol for communication networks, in: Proc. IEEE INFOCOM 97 Kobe, Japan, 1997, pp. 738–745.
[12] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997, 504 p.
[13] T. Kwon, M. Kang, S. Jung, J. Song, An improvement of the password-based authentication protocol (K1P) on security against replay attacks, IEICE Trans. Commun. E82-B (7) (July 1999) 991–997.

[14] T. Kwon, J. Song, Authentication key exchange protocols resistant to password guessing attacks, IEE Commun. 145 (5) (October 1998) 304–308.
[15] M. Steiner, G. Tsudik, M. Waidner, Refinement and extension of encrypted key exchange, Oper. Syst. Rev. 29 (3) (July 1995) 22–30.
[16] Y. Ding, P. Horster, Undetectable on-line password guessing attacks, Technical Report, TR-95-13-F, July 1995.
[17] J.T. Kohl, B.C. Neuman, T. Ts'o, The evolution of the kerberos authentication system, in: Distributed Open System, IEEE Comput. Soc. Press, 1994, pp. 78–94.
[18] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Trans. Inform. Theory IT-11 (November 1976) 644–654.
[19] R. Needham, M. Schroeder, Using encryption for authentication in large networks of computers, Commun. ACM 21 (12) (1978) 993–999.