IJITCE

ISSN 2347- 3657

International Journal of Information Technology & Computer Engineering

Email : ijitce.editor@gmail.com or editor@ijitce.com

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 11, Issue 1, 2023

Password-based authentication and key distribution protocolswith perfect forward secrecy

Prof. D. Raja, Mr. S. Sugavanam, Ms. R. Roopa, Mrs. M. Indra Priya
Professor [1], Associate Professor [2,4] Assistant Professor [3]

raja.d@actechnology.in, sugavanam.s@actechnology.in, roopa.r@actechnology.in, indrapriya.m@actechnology.in

Department of CS & BS, Arjun College of Technology, Thamaraikulam, Coimbatore-Pollachi Highway, Coimbatore, Tamilnadu-642 120

Abstract

In order to provide its services in an open networking environment, a workstation often has to identify its legitimate users. One effective method is Kerberos, which uses a trusted third-party authentication server to confirm the identity of users. Unfortunately, Kerberos is vulnerable to password guessing attacks since it requires users to utilise strong cryptographic secrets for authentication, which may be inconvenient if users use weak passwords. Our main emphasis in this work is on a system that allows users to use passwords that are simple to remember. When developing a system for authentication and key distribution, it is crucial to keep in mind both password guessing attacks and perfect forward secrecy, or PFS for short. We identify seven classes of perfect forward secrecy based on the capacity to secure the client's password, the application server's secret key, and the authentication server's private key. Protocols that achieve class-1, class-3, and class-7 are given special attention owing to their hierarchical linkages. Afterwards, in order to guarantee absolute forward secrecy for these three categories, we provide three safe methods of authentication and key distribution. All of these techniques work well to prevent guessing and replay attacks on passwords that users have picked with less than ideal care.

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

1. Introduction

Keeping data safe when using unsecured connections is a major concern in modern distributed computing. This is why the distribution of secret keys and user authentication have emerged as the most critical security services for communication networks. Protocols for key distribution and authentication are so essential in distributed systems.

Due to its inherent simplicity—passwords may be freely chosen and remembered by the user—password-based mechanisms have long predominated in user authentication. Password guessing attacks are common because human users often select passwords that are simple to remember. Servers and other non-human entities may circumvent password guessing attacks by authenticating themselves directly using strong cryptographic secrets.

ISSN 2347–3657
Volume 11, Issue 1, 2023

Numerous communication systems have adopted and implemented various authentication and key distribution techniques in the last few years. A shared session key may be established by public communications, as described by Diffie and Hellman [18]. To accomplish authentication with encryption, Needham and Schroeder [19] put up a point protocol. The acronym EKE stands for "Encrypted Key Exchange," a technique introduced in 1992 by Bellovin and Merritt [1]. Key exchange methods and two-party authentication have it as their defining feature [1–7]. By providing the adversary with inadequate information, EKE is able to withstand guessing attempts. Since EKE also executes a key exchange, once authentication is accomplished, both parties may encrypt their messages. Alternatively, in a three-party context, Gong, Lomas, Needham, and Saltzer [8] put forth a protocol known as the GLNS protocol, whereby two clients (users) create a session key via an authentication server. To ensure that messages are up-to-date, the protocol makes use of timestamps. In order to withstand offline password guessing assaults, the system generates a vast search space utilising nonces and confounders. Gong subsequently suggested an improved version of the GLNS protocol [9] that eliminates the need for timestamps and drastically cuts down on message transfers. An additional approach that is resistant to offline password guessing and replay attacks was suggested by Keung and Siu [10]. An additional technique for the distribution of keys and mutual authentication was suggested by Kwon, Kang, and Song [11]. Both protocols use the idea of a one-way hash function and a one-time pad to cut down on calculation costs. The majority of the existing research on key distribution methods and three-party authentication has been on the client-client-server paradigm, in which two users (clients) create a session key via an authentication server. But imagine a networked, open environment where users at workstations want to access services hosted on servers spread out over the network. Here, a central server (the authentication server) is responsible for authorising clients to access certain services hosted by other servers (the application servers). Two servers and a client make up this model. A client-server-server model is used to differentiate this from the current approach. It is not possible to have faith in a workstation's ability to authenticate its users to network services in this setting. Specifically, there are three dangers that might occur:

An unauthorised user may take over another user's account by gaining access to their workstation. A user may impersonate another user's workstation by changing its network address, making it seem as if queries are coming from that user's machine.

• An eavesdropper might potentially acquire access to a server or cause disruptions in operations by listening in on exchanges and launching a replay attack.

If any of these things happen, an unauthorised user can potentially acquire access to sensitive information and services. A centralised authentication server is supplied to authenticate users to servers and servers to users, instead than implementing complex authentication protocols for each server. There are a lot of uses for this kind of setting. The famous Kerberos protocol is a common example utilised in this kind of environment [17]. Kerberos, on the other hand, is vulnerable to password guessing attacks since it requires a strong cryptographic secret for user authentication. In this research, we zero in on a scenario where users use simple, memorable passwords. When developing a system for authentication and key distribution, it is crucial to keep in mind both password guessing attacks and perfect forward secrecy, or PFS for short. Seven types of perfect forward secrecy are defined according to the capacity to safeguard the client's password, the application server's secret key, and the authentication server's private key. Because of their dominance in security from low to high level, class-1, class-3, and class-7 PFS are of particular relevance to us. In addition, we provide three protocols that are suitable for this setting, secure against different types of attacks, and individually accomplish the aforementioned PFS classes.

What follows is an outline of the rest of the paper. Section 2 provides a concise overview of the specification of notations and security needs. Section 3 presents a class-1 PFS procedure. In Section 4, we provide a procedure for class-3 PFS. Section 5 proposes a procedure for class-7 PFS. We evaluate these three brand-new methods alongside several established ones in Section 6. In Section 7, we draw a conclusion to this work.

2. Notations and security requirements

2.1. Notations

The notations in Table 1 are used throughout this paper.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 11, Issue 1, 2023

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^r	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)
$[info]_K$	Symmetric encryption of "info" with key K
$\{info\}_K$	Asymmetric encryption of "info" with public key K

2.2. Security requirements

This article defines seven types of perfect forward secrecy and discusses some well-known vulnerabilities, such as replay attacks and password guessing attacks.

Intruders trying to guess passwords:

There are two main categories of password guessing attacks:

1) Attacks using online password guessing: In these attacks, the attacker attempts to bypass the authentication server's verification process by using a password that has already been guesses. Typically, the authentication server will be able to identify this kind of assault if it notices a pattern of persistent authentication failures. (2) Attacks that involve guessing passwords offline: An intruder records and retains communication messages that occur during a protocol. After that, using the collected data in an offline fashion, he or she attempts to guess a weak password and then checks to see whether the guess was right. In most cases, the only way to stop this kind of attack is to make sure the protocol isn't designed to allow the attacker to use any verifiable information to check whether their password guess is right.

Return to attack:

Here, the bad guy attempts to repeat messages that he or she has partly or fully gotten from earlier conversations. The protocol is considered replay-attack susceptible if an attacker can impersonate other users or reveal important information that might be used for additional deceptions via guessing attacks, known-plaintext attacks, or other means of cryptographic analysis.

Utmost confidentiality in advance (PFS):

If an attacker cannot use the disclosed password to retrieve the session keys of previous sessions in a two-party environment, then the password-based protocol is considered perfect forward secure [5,8,16]. In a three-party context, we categorise instances of perfect forward secrecy based on the likelihood that the client's password, application server's secret key, and authentication server's private key are disclosed. We establish seven types of perfect forward secrecy depending on the capacity to safeguard the client's password, the application server's secret key,and the authentication server's private key. We display these seven types of perfect forward secrecy in Table 2. Assuming the application server's and authentication server's private keys remain secret, a protocol offering class-1 PFS would prevent an attacker from obtaining session keys from prior communications, even if the genuine password were to be revealed.

Take into consideration that the likelihood of the client's password being leaked is much greater than that of the application server's secret key, and that of the application server's secret key being leaked is significantly higher than that of the authentication server's private key. Our interest in developing protocols that meet the needs of classes 1, 3, and 7 stems from the hierarchical relationships that exist among these PFS types. The following provide extensive explanations of class-1 PFS, class-3 PFS, and class-7 PFS.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 11, Issue 1, 2023

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

In this part, we provide a protocol for class-1 PFS that is both efficient and secure for authentication and key distribution. In our protocol, three main players work together: an application server B that offers services to clients, a client A that makes service requests to the application server, and an authentication server S that verifies the client and gives them a common session key to use with the application server.

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) $A \rightarrow S: A, \{A, B, P_A, ra\}_{K_S}$:
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.

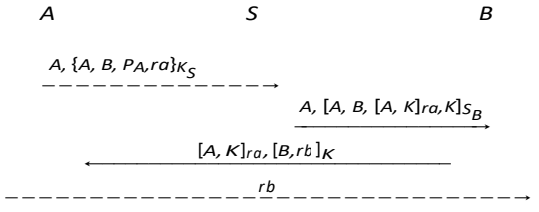

Fig. 1. A protocol providing PFS with low security.

(2) $S \rightarrow B$: $A, [A, B, [A, K]_{ra}, K]_{SB}$:
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 $[A, B, [A, K]_{ra}, K]_{SB}$, and transmit it to B. Note that the value ra also acts as a one-time key.

(3) $B \rightarrow A$: $[A, K]_{ra}, [B, rb]_K$:
The application server B first decrypts the message $[A, B, [A, K]_{ra}, K]_{SB}$ 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) $A \rightarrow B$: 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*

The goal of this protocol is to prevent the password from being compromised by an attacker, regardless of their level of competence. A failed guess may be recognised and recorded if an attacker tries to use a guess password in an online transaction. Therefore, our system is secure against attacks that use online guessing. Only A's status may be authenticated in Message 1 using A's password, taking into consideration an offline guessing attack. No data that can be verified includes it. Because of this, it is hard for an attacker to validate their password guess without knowledge of the random number ra. Hence, our protocol is secure against offline attacks that attempt to guess passwords.

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 $A, B, A, K_{ra}, K_{SB}, A, K_{ra}, B, rb_K$. 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 sessionkey K^r 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 1to 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) $A \rightarrow B$: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.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 11, Issue 1, 2023

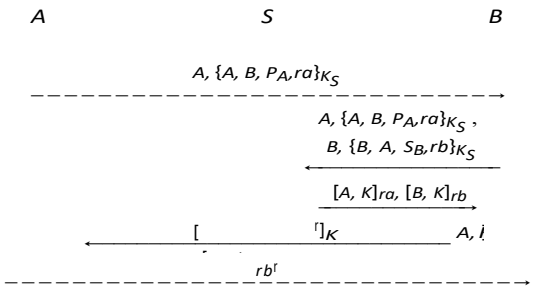

Fig. 2. A protocol providing PFS with middle security.

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

(5) $A \rightarrow B$: rb^r:
In step 5, the client A decrypts message $[A, K]_{ra}$ with ra and gets the common key K. Then, he decrypts $[B, rb^r]_K$, checks the validity of K, and sends the response value rb^r 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^r 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.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 11, Issue 1, 2023

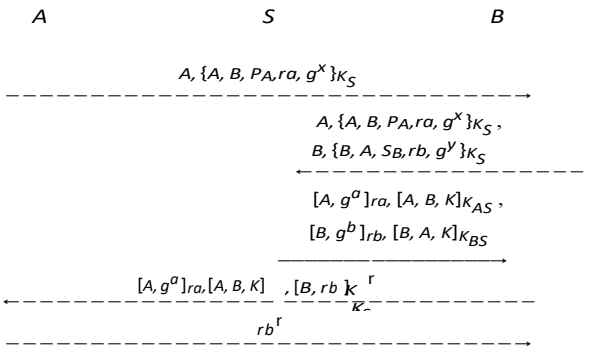

Fig. 3. A protocol providing PFS with high security.

The details are described as follows:

(1) $A \rightarrow B$: $A, \{A, B, P_A, ra, g^x\}_{K_S}$:
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) $B \rightarrow S$: $A, \{A, B, P_A, ra, g^x\}_{K_S}, B, \{B, A, S_B, rb, g^y\}_{K_S}$:
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) $S \rightarrow B$: $[A, g^a]_{ra}, [A, B, K]_{K_{AS}}, [B, g^b]_{rb}, [B, A, K]_{K_{BS}}$:
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) $B \rightarrow A$: $[A, g^a]_{ra}, [A, B, K]_{K_{AS}}, [B, rb^r]_K$:
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 $[B, A, K]_{KBS}$ with K_{BS} and gets the common key K. Then, B uses rb^r as a challenge value, encrypts B, rb^r with the common key K and sends $[A, g^a]_{ra}, [A, B, K]_{K_{AS}}, [B, rb^r]_K$ to the client A.

(5) $A \rightarrow B$: rb^r:
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 $[B, rb^r]_K$, checks the validity of K, and sends the response value rb^r 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

6 This protocol does not require the client's password to authenticate the state of A. This means an attacker can't tell whether his guess is correct since there is no way to verify the data. Password guessing attacks cannot compromise our protocol.

6.2.1. Replay attacks

7 The inability of the server to determine the freshness of messages allows an attacker to replay older ones, such as Messages 1 and 2. Nevertheless, using the knowledge provided by S, he cannot deduce the password or compromise a future session key. Being protected against message replay attacks, our protocol is a go-go.

.

ISSN 2347–3657
Volume 11, Issue 1, 2023

7.2.1. 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

User and application server authentication using authentication server is the main topic of this article, which focusses on a three-party context (client-server-server paradigm). Since no specific protocol has been developed for this kind of setup, one workaround would be to utilise a client-server approach, which involves placing one of the clients in the role of application server. Hence, we evaluate our three recently suggested protocols—the low PFS protocol, the medium PFS protocol, and the high PFS protocol—against a number of popular conventional three-party protocols, such as the optimum GLNS protocol [9], the enhanced K1P protocol [13], and the extension of Otway-Rees protocol [12]. Several aspects are our primary concern, including the kinds of shared secrets, the number of steps, the amount of random numbers, the quantity of symmetric and asymmetric encryption operations, and so on. Other comparisons, such as the total volume of data exchanged, are not taken into consideration since these things vary for various security levels. The comparative findings are shown in Table 3.
There is no PFS class to which the Otway-Rees protocol extension belongs. Our medium PFS protocol (class-3 PFS) uses the same amount of steps and asymmetric encryption operations as the optimum GLNS and enhanced K1P protocols, but uses less random numbers and fewer symmetric encryption operations overall. Among these class-3 PFS protocols, our medium PFS treatment outperforms the others, according to the data. Furthermore, unlike competing protocols, our high PFS methodology is able to provide class-7 PFS.
One may regard one of the clients as the application server by applying an existing three-party client-client-server protocol with class-3 PFS to the client-server-server model with class-3 PFS, as shown in, for example, [9] and [13]. In place of a password, the application server may make use of a strong key, also known as a secret key. It seems that new protocols for the client-server-server paradigm are not necessary in light of this. But there are three ways in which we might advocate for new protocols inside the client-server-server paradigm. The first step in protecting against client-side password guessing attacks is to implement a client-client-server communication. One side of password guessing attacks may always be prevented if one of the clients, the application server, can utilise a strong key as its secret. Because of this, we think a client-server-server protocol may potentially have a lower cost than a client-client-server protocol. Table 3 shows that this is the case when we compare our medium PFS protocol to the best GLNS protocol [9] and the enhanced K1P protocol [13]. Secondly, our low PFS protocol can do what is needed for a client-server-server communication with class-1 PFS. Our medium PFS protocol, the optimum GLNS protocol, and the enhanced K1P protocol can all achieve class-1 PFS (since class-3 PFS is equivalent to class-1 PFS), but they aren't optimised for class-1 PFS and are thus inefficient in terms of cost. Finally, the majority of the currently available client-client-server protocols are able to achieve class-3 PFS. Since the client-server-server architecture does not support client-client-server protocols, we cannot use them to provide class-7 PFS. A new protocol that is up to snuff in terms of security will have to be designed instead. In conclusion, the comparison table clearly shows that the custom protocols suggested in this study are far better than just changing the current client-client-server protocols.

7. Conclusions

This article discusses an open distributed environment where clients access application server services via an authentication server. When developing an authentication and key distribution technique for this setting, perfect forward secrecy should be a top priority. We identify seven types of authentication servers based on their capacity to secure client passwords, application server secrets, and authentication server private keys.

ISSN 2347–3657
Volume 11, Issue 1, 2023

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.

ISSN 2347–3657
Volume 11, Issue 1, 2023

[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.