4-2006

A practical password-based two-server authentication and key exchange system

Yanjiang YANG
Singapore Management University, yjyang@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Feng Bao
Singapore Management University, fbao@smu.edu.sg

DOI: https://doi.org/10.1109/TDSC.2006.16

Citation

YANG, Yanjiang; DENG, Robert H.; and Bao, Feng. A practical password-based two-server authentication and key exchange system. (2006). *IEEE Transactions on Dependable and Secure Computing*. 3, (2), 105-114. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/1190

Published in IEEE Transactions on Dependable and Secure Computing, 2006 April, Volume 3, Issue 2, Pages 105-114
https://doi.org/10.1109/TDSC.2006.16

A Practical Password-Based Two-Server Authentication and Key Exchange System

Yanjiang Yang, Robert H. Deng, *Senior Member*, *IEEE*, and Feng Bao

Abstract—Most password-based user authentication systems place total trust on the authentication server where cleartext passwords or easily derived password verification data are stored in a central database. Such systems are, thus, by no means resilient against offline dictionary attacks initiated at the server side. Compromise of the authentication server by either outsiders or insiders subjects all user passwords to exposure and may have serious legal and financial repercussions to an organization. Recently, several multiserver password systems were proposed to circumvent the single point of vulnerability inherent in the single-server architecture. However, these multiserver systems are difficult to deploy and operate in practice since either a user has to communicate simultaneously with multiple servers or the protocols are quite expensive. In this paper, we present a practical password-based user authentication and key exchange system employing a novel two-server architecture. Our system has a number of appealing features. In our system, only a front-end *service server* engages directly with users while a *control server* stays behind the scene; therefore, it can be directly applied to strengthen existing single-server password systems. In addition, the system is secure against offline dictionary attacks mounted by either of the two servers.

Index Terms—Password system, password verification data (PVD), user authentication, key exchange, offline dictionary attack.

1 Introduction

PASSWORD-BASED user authentication systems are low cost and easy to use. A user only needs to memorize a short password and can be authenticated anywhere, anytime, regardless of the types of access devices he/she employs. By and large, password has been the most pervasive user authentication means since the advent of computers and is still gaining popularity even in the presence of several alternative strong authentication approaches, e.g., digital signature and biometrics [15]. The reasons are straightforward: Password authentication requires no dedicated device, which is of special importance as users are becoming increasingly roaming nowadays. While smartcards or similar handheld devices offer good portability for storing secret signing keys in digital signature generation, they inevitably require supporting infrastructure (software, hardware, and PKI) to work upon; moreover, safety of the physical token itself is a concern: Theft or loss of the token not only risks disclosing the secrets inside but also disables the authentication functionality. As far as biometrics are concerned, first, they also exclusively rely on costly underlying hardware and software infrastructure; second, biometrics are typically created as an authentication means for physical access control and have not yet matured enough to support online services; third, there exist intense debates and suspicion that biometrics may compromise individual privacy if the biometrics data are leaked or abused [23].

However, the use of passwords has intrinsic weaknesses. It is a well-known problem that human-user-chosen passwords are inherently weak since most users choose short and easy to remember passwords. In particular, passwords are normally drawn from a relatively small *dictionary*, so it allows for brute-force *dictionary attacks*, where an attacker enumerates every possible password in the dictionary to determine the actual password. Dictionary attacks can be mounted *online* or *offline*. In an online dictionary attack, attackers attempt to log in to a server by trying all possible passwords from the dictionary until they find a correct one. In an offline dictionary attack, attackers record a past successful login session between a user and a server and then check all the passwords in the dictionary against the login transcript. Online dictionary attacks can be easily thwarted at the system level by limiting the number of unsuccessful login attempts made by a user. In contrast, offline dictionary attacks are notoriously harder to deal with. As a result, tremendous effort has been dedicated to countering offline dictionary attacks in password systems.

1.1 Related Work

It is a proven fact that *public key techniques* (e.g., exponentiations in a multiplicative group) are absolutely necessary to make password systems secure against offline dictionary attacks, whereas the involvement of *public key cryptosystems* under a PKI (e.g., public key encryption and digital signature schemes) is not essential [13]. This observation differentiates two separate approaches to the development of secure password systems: combined use of a password and public key cryptosystem under a PKI, and a password-only approach. The former takes into account the asymmetry of capabilities between users and servers, so a user only uses a password while the server has a public/private key pair at its disposal. Examples of such public key-assisted password systems include [11], [13], [6]. In these

- Y. Yang and R.H. Deng are with the School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902. E-mail: {yjyang, robertdeng}@smu.edu.sg.
- F. Bao is with the Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613. E-mail: baofeng2i2r.a-star.edu.sg.

Manuscript received 29 Mar. 2005; revised 23 Dec. 2005; accepted 15 Feb. 2006; published online 4 May 2006.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0042-0305.

systems, the use of public keys entails the deployment and maintenance of a PKI for public key certification and adds to users the burden of checking key validity. To eliminate this drawback, password-only protocols (password authenticated key exchange or PAKE) have been extensively studied, e.g., [1], [4], [5], [19], [20], [7]. The PAKE protocols do not involve any public key cryptosystem under a PKI and, therefore, are much more attractive for real-world applications. We believe that any use of public key cryptosystem under a PKI in a password authentication system should be avoided since, otherwise, the benefits brought by the use of password would be counteracted to a great extent.

Most of the existing password systems were designed over a single server, where each user shares a password or some password verification data (PVD) with a single authentication server (e.g., [1], [4], [5], [6], [7], [16], [11], [13], [19], [20]). These systems are essentially intended to defeat offline dictionary attacks by outside attackers and assume that the sever is completely trusted in protecting the user password database. Unfortunately, attackers in practice take on a variety of forms, such as hackers, viruses, worms, accidents, misconfigurations, and disgruntled system administrators. As a result, no security measures and precautions can guarantee that a system will never be penetrated. Once an authentication server is compromised, all the user passwords or PVD fall in the hands of the attackers, who are definitely effective in offline dictionary attacks against the user passwords. To eliminate this single point of vulnerability inherent in the single-server systems, password systems based on multiple servers were proposed. The principle is distributing the password database as well as the authentication function to multiple servers so that an attacker is forced to compromise several servers to be successful in offline dictionary attacks.

The system in [10], believed to be the first multiserver password system, splits a password among multiple servers. However, the servers in [10] need to use public keys. An improved version of [10] was proposed in [14], which eliminates the use of public keys by the servers. Further and more rigorous extensions were due to [21] and [22], where the former built a t-out-of-n threshold PAKE protocol and provided a formal security proof under the random oracle model [8] and the latter presented two provably secure threshold PAKE protocols under the standard model. While the protocols in [21] and [22] are theoretically significant, they have low efficiency and high operational overhead. In these multiserver password systems, either the servers are equally exposed to the users and a user has to communicate in parallel with several or all servers for authentication, or a gateway is introduced between the users and the servers. We shall further discuss the disadvantages of the multiserver models in Section 2.

Recently, Brainard et al. [3] proposed a two-server password system in which one server exposes itself to users and the other is hidden from the public. While this two-server setting is interesting, it is not a password-only system: Both servers need to have public keys to protect the communication channels from users to servers. As we have stressed earlier, this makes it difficult to fully enjoy the benefits of a password system. In addition, the system in [3] only performs unilateral authentication and relies on the Secure Socket Layer (SSL) to establish a session key between a user and the front-end server. Subsequently, Yang et al. [24] extended and tailored this two-server system to the context of federated enterprises, where the back-end server is managed by an enterprise headquarter and each affiliating organization operates a front-end server. An improvement made in [24] is that only the back-end server holds a public key. Nevertheless, the system in [24] is still not a password-only system. We notice that the two-server system presented in [17] does not follow the two-server paradigm in [3], [24], but is a special case of the earlier multiserver systems (see Section 2 for details).

1.2 Our Contribution

We continue the line of research on the two-server paradigm in [3], [24], whereas we extend the model by imposing different levels of trust upon the two servers, and adopt a very different method at the technical level in the protocol design. As a result, we propose a *practical* two-server password authentication and key exchange system that is secure against offline dictionary attacks by servers when they are controlled by adversaries. Our system is a password-only system in the sense that it requires no public key cryptosystem and, thus, no PKI. This makes our system very attractive considering PKIs are proven notoriously expensive to deploy in real world. Moreover, our proposed system is particularly suitable for resource-constrained users due to its efficiency in terms of both computation and communication. We generalize the basic two-server model to an architecture of a single back-end server supporting multiple front-end servers and envision interesting applications in federated enterprises.

1.3 Organization

In Section 2, we discuss different server models for password systems and specify the two-server architecture upon which ours is built. We then present our two-server password authentication and key exchange protocols in Section 3. In Section 4, we describe applications and extension of the proposed system, followed by some discussions in Section 5. Finally, we draw concluding remarks and give future work in Section 6.

2 The Two-Server Architecture

Password systems are normally built over the following four types of architectures shown in Fig. 1.

The first type is the *single-server model* given in Fig. 1a, where a single server is involved and it keeps a database of user passwords. As mentioned earlier, most of the existing password systems follow this single-server model, but the single server results in a single point of vulnerability in terms of offline dictionary attacks against the user password database.

The second type is the *plain multiserver model* depicted in Fig. 1b, in which the server side comprises multiple servers for the purpose of removing the single point of vulnerability; the servers are equally exposed to users and a user has to communicate in parallel with several or all servers

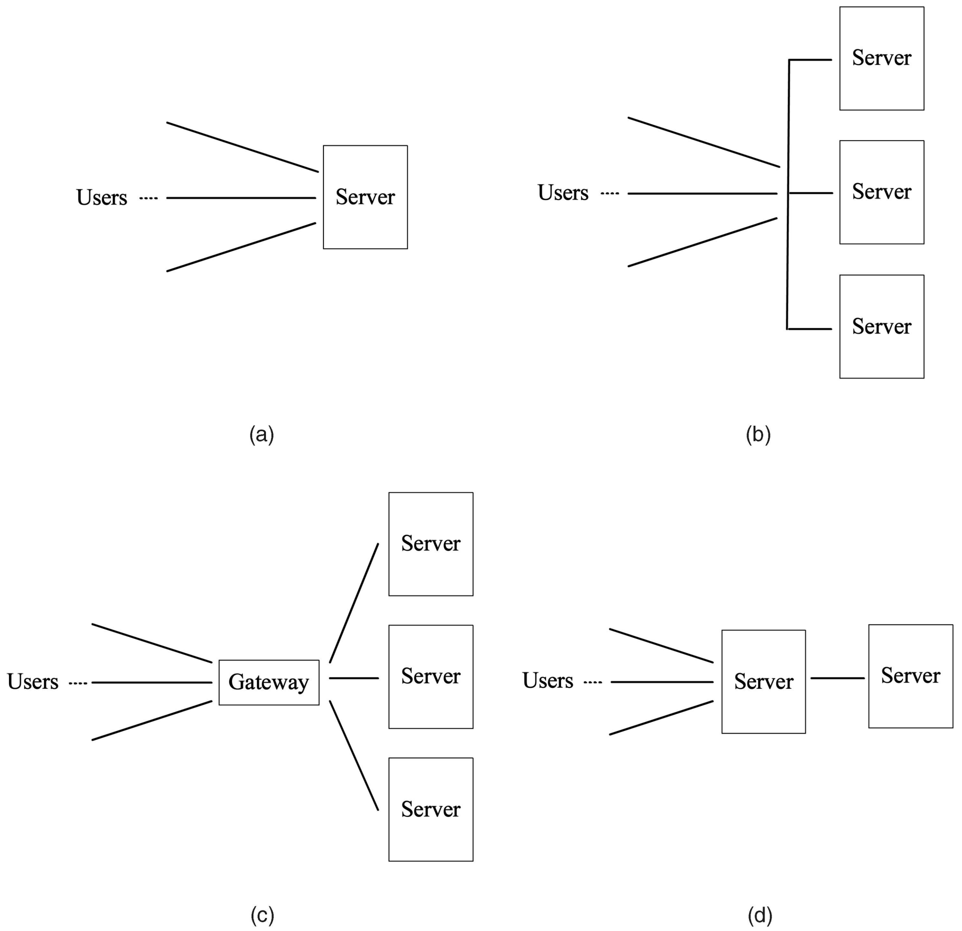


Fig. 1. Models of password systems. (a) Single-server model. (b) Plain multiserver model. (c) Gateway augmented multiserver model. (d) Two-server model.

for authentication. Clearly, the main problem with the plain multiserver model is the demand on communication bandwidth and the need for synchronization at the user side since a user has to engage in simultaneous communications with multiple servers. This may cause problems to resource-constrained mobile devices such as hand phones and PDAs. The systems in [10], [14], [21] and one of the two protocols in [22] assume this model.

The third type is the *gateway augmented multiserver model* shown in Fig. 1c, where a gateway is positioned as a relaying point between users and servers and a user only needs to contact the gateway. Apparently, the introduction of the gateway removes the demand of simultaneous communications by a user with multiple servers as in the plain multiserver model. However, the gateway introduces an additional layer in the architecture, which appears "redundant" since the purpose of the gateway is simply to relay messages between users and servers, and it does not in any way involve in service provision, authentication, and other security enforcements. From security perspective, more components generally imply more points of vulnerabilities. Protocols based on the gateway augmented multiserver model include [17] and [22].

The fourth type is the *two-server model* (outlined in Fig. 1d), that comprises two servers at the server side, one of which is a *public server* exposing itself to users and the other of which is a *back-end server* staying behind the scene; users contact only the public server, but the two servers work together to authenticate users. It is important to note the essential differences between the two-server model and the earlier multiserver models: 1) In the two-server model, a user ends up establishing a session key only with the public server, and the role of the back-end server is merely to assist the public server in user authentication, while in the multiserver models, a user establishes a session key (either different or the same) with each of the servers. For exactly this reason, we view the two-server system in [17] as a special case of the gateway augmented multiserver model of two servers. 2) From a security point of view, servers in the multiserver models are equally exposed to outside attackers (recall that the gateway in the gateway augmented multiserver model does not enforce security), while in the two-server model, only the public server faces such a problem. This clearly improves the server side security and in turn the overall system security in the two-server model.

Another observation on the two-server model is that we can assume different levels of trust upon the two servers with respect to outside attackers. Specifically, the back-end server is more trustworthy than the public server. This is logical since the back-end server is located in the back-end and is hidden from the public, and it is thus less likely to be attacked. Further justifications with respect to inside

TABLE 1
Notations

Q, p, q	three large primes such that $Q = 2p + 1$ and $p = 2q + 1$.
g_1, g_2	$g_1, g_2 \in QR_p$ are of order q and the discrete logarithms to each other are not known, where QR_p is the group of quadratic residues modulo p.
g_3	$g_3 \in QR_Q$ is of order p.
π	a user's password.
$h(.)$	a cryptographic hash function modelled as the random oracle [8].
$\mathcal{U}$, $\mathcal{SS}$, $\mathcal{CS}$	identities of user, service sever and control server, respectively.

attackers come from the application and extension of our proposed two-server system in Section 4. As we will see shortly, this assumption is well exploited in our design of the password authentication and key exchange protocols.

It is clear that the two-server model has successfully eliminated drawbacks in the plain multiserver model (i.e., simultaneous communications between a user and multiple servers) and the gateway augmented multiserver model (i.e., redundancy) while allowing us to distribute user passwords and the authentication functionality to two servers in order to eliminate a single point of vulnerability in the single-server model. As a result, the two-server model appears to be a sound model for practical applications. However, as we pointed out earlier, the existing systems upon the two-server model such as [3], [24] do not suffice; we are thus motivated to present a password-only system over the two-server model. In particular, in our system, the public server acts as a *service server* that provides application services, while the back-end server is a *control server* whose sole purpose is to assist the service server in user authentication (the service server, of course, also participates in user authentication). This enforces clear *separation of duty* in our system. We highlight that in the plain multiserver model and the gateway augmented multiserver model, several or all servers equally participate in service provision as well as user authentication, which is implied by the fact that a user negotiates a session key with each server. We also generalize the two-server model to an architecture that a control server supports multiple service servers (see Section 4).

3 Two-Server Password Authentication Protocols

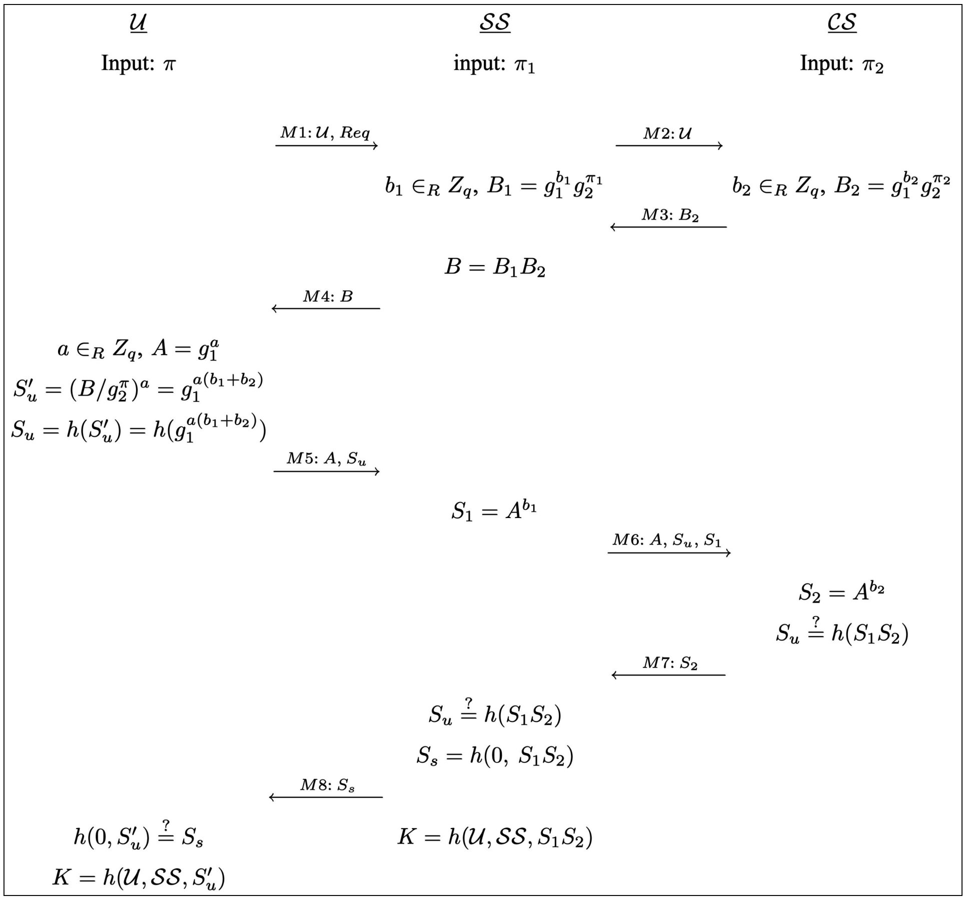
In this section, we elaborate on our proposed password authentication and key exchange protocols upon the two-server model. In particular, we first present a basic protocol, and examine its security; we then show how to circumvent the weaknesses contained in the basic protocol by presenting an improved protocol. Compared to the systems in [3], [24], we completely avoid the use of a public key cryptosystem at the server side. Our protocols are quite efficient in terms of both communication and computation. For ease of reference, notations that are used below are listed in Table 1.

3.1 System Model

Three types of entities are involved in our system, i.e., users, a service server ($\mathcal{SS}$) that is the public server in the two-server model, and a control server ($\mathcal{CS}$) that is the back-end server. In this setting, users only communicate with $\mathcal{SS}$ and do not necessarily know $\mathcal{CS}$. For the purpose of user authentication, a user $\mathcal{U}$ has a password which is transformed into two long secrets, which are held by $\mathcal{SS}$ and $\mathcal{CS}$, respectively. Based on their respective shares, $\mathcal{SS}$ and $\mathcal{CS}$ together validate users during user login.

We assume the following security model: $\mathcal{CS}$ is controlled by a *passive adversary* and $\mathcal{SS}$ is controlled by an *active adversary* in terms of offline dictionary attacks to user passwords, but they do not collude (otherwise, it equates the single-server model). By definition (e.g., [12]), a passive adversary follows honest-but-curious behavior, that is, it honestly executes the protocol according to the protocol specification and does not modify data, but it eavesdrops on communication channels, collects protocol transcripts and tries to derive user passwords from the transcripts; moreover, when an passive adversary controls a server, it knows all internal states of knowledge known to the server, including its private key (if any) and the shares of user passwords. In contrast, an active adversary can act arbitrarily in order to uncover user passwords. Besides, we assume a secret communication channel between $\mathcal{SS}$ and $\mathcal{CS}$ for this basic protocol. We shall discuss how to remove this assumption in the improved protocol.

We stress that this security model exploits the different levels of trust upon the two servers. As discussed earlier, this clearly holds with respect to outside attackers. As far as inside attackers are concerned, justifications come from our application and generalization of the system to the architecture of a single control server supporting multiple service servers, where the control server affords and deserves enforcing more stringent security measurements against inside attackers (please refer to Section 4 for details). Notice that the assumption we make here is already weaker than that in [24], where the back-end server is strictly passive and is not allowed to eavesdrop on communication channels, while $\mathcal{CS}$ in our setting is allowed for eavesdropping. We believe this weakening is important and more realistic since eavesdropping is practically easy and insidious.



$\underline{\mathcal{U}}$
$\underline{\mathcal{SS}}$
$\underline{\mathcal{CS}}$
Input: π
input: π_1
Input: π_2
$M1$: $\mathcal{U}$, Req
$M2$: $\mathcal{U}$
$b_1 \in_R Z_q$, $B_1 = g_1^{b_1} g_2^{\pi_1}$
$b_2 \in_R Z_q$, $B_2 = g_1^{b_2} g_2^{\pi_2}$
$M3$: B_2
$B = B_1 B_2$
$M4$: B
$a \in_R Z_q$, $A = g_1^a$
$S'_u = (B/g_2^{\pi})^a = g_1^{a(b_1+b_2)}$
$S_u = h(S'_u) = h(g_1^{a(b_1+b_2)})$
$M5$: A, S_u
$S_1 = A^{b_1}$
$M6$: A, S_u, S_1
$S_2 = A^{b_2}$
$S_u \overset{?}{=} h(S_1 S_2)$
$M7$: S_2
$S_u \overset{?}{=} h(S_1 S_2)$
$S_s = h(0,\ S_1 S_2)$
$M8$: S_s
$h(0, S'_u) \overset{?}{=} S_s$
$K = h(\mathcal{U}, \mathcal{SS}, S_1 S_2)$
$K = h(\mathcal{U}, \mathcal{SS}, S'_u)$

Fig. 2. A basic password authentication and key exchange protocol.

3.2 High-Level Description

Central to our protocol design is the defense against offline dictionary attacks by the servers when they are controlled by adversaries. The intuition is to "harden" a user's short password π into two long shares π_1 and π_2 in such a way that they are no longer subject to offline dictionary attacks, and then distribute them to the two servers. As a result, an attacker cannot succeed in offline dictionary attacks without grabbing both shares by compromising both servers. During user login, the control server $\mathcal{CS}$ using its share π_2 assists the service server $\mathcal{SS}$ using π_1 in user authentication. More specifically, in an out-of-band *user registration* phase, user $\mathcal{U}$ splits his password π into two long random secrets π_1 and π_2 and registers them to $\mathcal{SS}$ and $\mathcal{CS}$, respectively, where $\pi_1 + \pi_2 = \pi$. During *authentication*, $\mathcal{U}$ using π and $\mathcal{SS}$ using π_1 authenticate each other and negotiate a secret session key, with the help of $\mathcal{CS}$ using π_2.

3.3 User Registration

In any password system, to enroll as a legitimate user in a service, a user must beforehand register with the service provider by establishing a shared password with the provider. In our system, $\mathcal{U}$ needs to register not only to the service provider $\mathcal{SS}$ but also to the control server $\mathcal{CS}$. Let us suppose $\mathcal{U}$ has already successfully identified himself to $\mathcal{SS}$, e.g., by showing his identification card, $\mathcal{U}$ splits his password π into two long random numbers $\pi_1 \in_R Z_q$ and $\pi_2 \in_R Z_q$ such that $\pi_1 + \pi_2 = \pi \pmod q$, where q is defined in Table 1. $\mathcal{U}$ then registers in a secure manner π_1 and π_2 to $\mathcal{SS}$ and $\mathcal{CS}$, respectively. $\mathcal{SS}$ stores the account information ($\mathcal{U}$, π_1) to its secret database, and $\mathcal{CS}$ stores ($\mathcal{U}$, π_2) to its secret database. In case $\mathcal{CS}$ supports multiple servers, it stores ($\mathcal{U}$, π_2, $\mathcal{SS}$) to distinguish users associated with different servers. This completes the user registration phase. One may wonder how $\mathcal{U}$ registers π_2 to $\mathcal{CS}$ as $\mathcal{CS}$ is supposed hidden from $\mathcal{U}$. This actually is not a problem in practice: $\mathcal{U}$ can reach $\mathcal{CS}$ through out-of-band channels, such as postal mail. Indeed, imagine that a user enrolls in a bank; it is not strange at all that the user still needs to submit a secret to a higher authority of the bank so as to activate his account.

3.4 A Basic Password Authentication Protocol

Let p, q, g_1, g_2, and $h(.)$ be defined in Table 1. We outline the basic password authentication protocol in Fig. 2, which enables mutual authentication and key exchange between $\mathcal{U}$ and $\mathcal{SS}$. In the figure, we have omitted the modulo p notation for arithmetic operations, as this should be clear from the context.

To initiate a request for service, $\mathcal{U}$ sends his identity together with a service request Req to $\mathcal{SS}$ in $M1$. $\mathcal{SS}$ first relays the request to $\mathcal{CS}$ by sending the user ID in $M2$, and then selects a random number $b_1 \in_R Z_q$ and computes $B_1 = g_1^{b_1} g_2^{\pi_1} \pmod p$ using his password share π_1. Upon receiving $M2$, $\mathcal{CS}$ chooses a random number $b_2 \in_R Z_q$ and computes $B_2 = g_1^{b_2} g_2^{\pi_2} \pmod p$ using his password share π_2. $\mathcal{CS}$ then sends B_2 in $M3$ to $\mathcal{SS}$. Upon reception of B_2, $\mathcal{SS}$ computes and sends $B = B_1 B_2 (\bmod p)$ to $\mathcal{U}$ in $M4$. After receiving $M4$, $\mathcal{U}$ selects $a \in_R Z_q$, and computes $A = g_1^a (\bmod p)$, $S'_u = (B/g_2^{\pi})^a = g_1^{a(b_1+b_2)} (\bmod p)$ and $S_u = h(S'_u)$, respectively. $\mathcal{U}$ then sends A and S_u to $\mathcal{SS}$ in $M5$. Getting the message, $\mathcal{SS}$ computes $S_1 = A^{b_1} (\bmod p)$ and sends S_1, A and S_u to $\mathcal{CS}$ in $M6$. Upon receipt of $M6$, $\mathcal{CS}$ computes $S_2 = A^{b_2} (\bmod p)$ and checks whether $S_u \overset{?}{=} h(S_1 S_2) = h(g_1^{a(b_1+b_2)})$: If it holds, $\mathcal{CS}$ is assured of the authenticity of U, and continues the protocol by sending S_2 to $\mathcal{SS}$ in $M7$; otherwise, $\mathcal{CS}$ aborts the protocol.

Assuming $\mathcal{SS}$ receives S_2 in $M7$, it checks whether $S_u \overset{?}{=} h(S_1 S_2)$. If it holds, $\mathcal{SS}$ is convinced of the authenticity of $\mathcal{U}$. At this stage, both servers have authenticated $\mathcal{U}$. $\mathcal{SS}$ then computes and sends $S_s = h(0, S_1 S_2)$ to $\mathcal{U}$ in M_8 and afterward computes a session key $K = h(\mathcal{U}, \mathcal{SS}, S_1 S_2)$; otherwise, $\mathcal{SS}$ aborts the protocol. Upon receiving M_8, $\mathcal{U}$ checks if $h(0, S'_u) \overset{?}{=} S_s$. If it holds, $\mathcal{U}$ has validated the servers and then computes a session key $K = h(\mathcal{U}, \mathcal{SS}, S'_u)$; otherwise, $\mathcal{U}$ aborts the protocol.

3.5 Security Analysis

In what follows, we analyze the security of the basic protocol. Our analysis is based on the following Decisional Deffie-Hellman (DDH) assumption [2]:

DDH Assumption. *Let p, q be defined as in Table 1, and $g, h \in_R Z_p^*$ of order q, for every probabilistic polynomial time algorithm $\mathcal{A}$, the following condition is satisfied:*

$$Adv_G^{DDH}(\mathcal{A}) = |Pr[\mathcal{A}(g, h, g^r, h^r)] - Pr[\mathcal{A}(g, h, g^r, z)]| < \epsilon, \tag{1}$$

where $r \in_R Z_q^*$, $z \in_R QR_p$, and ϵ is a negligible function. Informally speaking, it is computationally intractable for $\mathcal{A}$ to distinguish between (g, h, g^r, h^r) and (g, h, g^r, z).

Recall that the primary goal of our protocol is to resist offline dictionary attacks by the two servers, where $\mathcal{CS}$ is controlled by a passive adversary and $\mathcal{SS}$ is controlled by an active adversary. Accordingly, we examine the protocol against $\mathcal{CS}$, $\mathcal{SS}$, and an active outside adversary that does not control any server, respectively.

Claim 1. *The protocol is robust against offline dictionary attacks by $\mathcal{CS}$ as a passive adversary.*

Proof. Intuitively, when $\mathcal{CS}$ is controlled by a passive adversary, it may eavesdrop on the communication channels to collect protocol transcript and try to launch offline dictionary attacks against the password of $\mathcal{U}$. Clearly, $\mathcal{CS}$ can obtain $B_1 = B/B_2 = g_1^{b_1} g_2^{\pi_1} (\bmod p)$ from $M4$. However, from B_1 alone, $\mathcal{CS}$ cannot learn anything of π_1 in an information theoretic sense. What remains relevant to $\mathcal{CS}$ for offline dictionary attacks are $[A = g_1^a,\ S_u = h((B/g_2^{\pi})^a)]$, and $[S_1 = A^{b_1},\ B_1 = g_1^{b_1} g_2^{\pi_1}]$. The first pair is clearly no easier than $[A = g_1^a,$ $S'_u = (B/g_2^{\pi})^a]$ for $\mathcal{CS}$ to deal with in terms of offline dictionary attacks; we thus suppose $\mathcal{CS}$ knows S'_u for ease of analysis. Note that $A = g_1^a \Rightarrow g_1 = A^{a^{-1}} (\bmod p)$ and $S'_u = (B/g_2^{\pi})^a \Rightarrow B/g_2^{\pi} = S_u'^{a^{-1}} (\bmod p)$. Under the DDH assumption, $\mathcal{CS}$ cannot distinguish between $[A,\ g_1 = A^{a^{-1}},\ S'_u,\ B/g_2^{\pi} = S_u'^{a^{-1}}]$ and $[A,\ A^{a^{-1}},\ S'_u,\ z]$, where $z \in_R QR_p$. This suggests that $\mathcal{CS}$ cannot get anything on π from the first pair. For the second pair, $B_1 = g_1^{b_1} g_2^{\pi_1} \Rightarrow B/g_2^{\pi_1} = g_1^{b_1} (\bmod p)$, and again under the DDH assumption, $\mathcal{CS}$ cannot distinguish between $[A,$ $S = A^{b_1},\ g_1,\ B/g_2^{\pi_1} = g_1^{b_1}]$ and $[A,\ A^{b_1},\ g_1,\ z]$. This shows that $\mathcal{CS}$ cannot learn anything on π_1 from the second pair. Consequently, $\mathcal{CS}$, controlled by a passive adversary, cannot be effective in offline dictionary attacks. This completes the proof. □

It is important to note that in the above analysis, we have implicitly assumed that $\mathcal{CS}$ does not know a, the discrete logarithm of A to the base g_1. However, were $\mathcal{CS}$ controlled by an active adversary, such an assumption would no longer hold since $\mathcal{CS}$ could simply impersonate $\mathcal{U}$, choose a, and compute $A = g_1^a (\bmod p)$. $\mathcal{CS}$ could also break the system if it were able to replace the original A from $\mathcal{U}$ with another one based on an a of its choice. In both cases, $\mathcal{CS}$ could find the password π by offline dictionary attacks. To see this, consider the second pair where $\mathcal{CS}$ knows $a = \log g_1^A\ (\bmod q)$ and the Diffie-Hellman quadruple $[A,\ S_1 = A^{b_1},\ g_1,\ B_2/g_2^{\pi_1} = g_1^{b_1}]$. It follows that $(B_2/g_2^{\pi_1})^a = (B_2/g_2^{\pi - \pi_2})^a = A^{b_1} = S_1$, so $\mathcal{CS}$ could try every possible password to determine the actual π with the knowledge of π_2. This explains at the technical level why $\mathcal{CS}$ is assumed to act as a passive adversary.

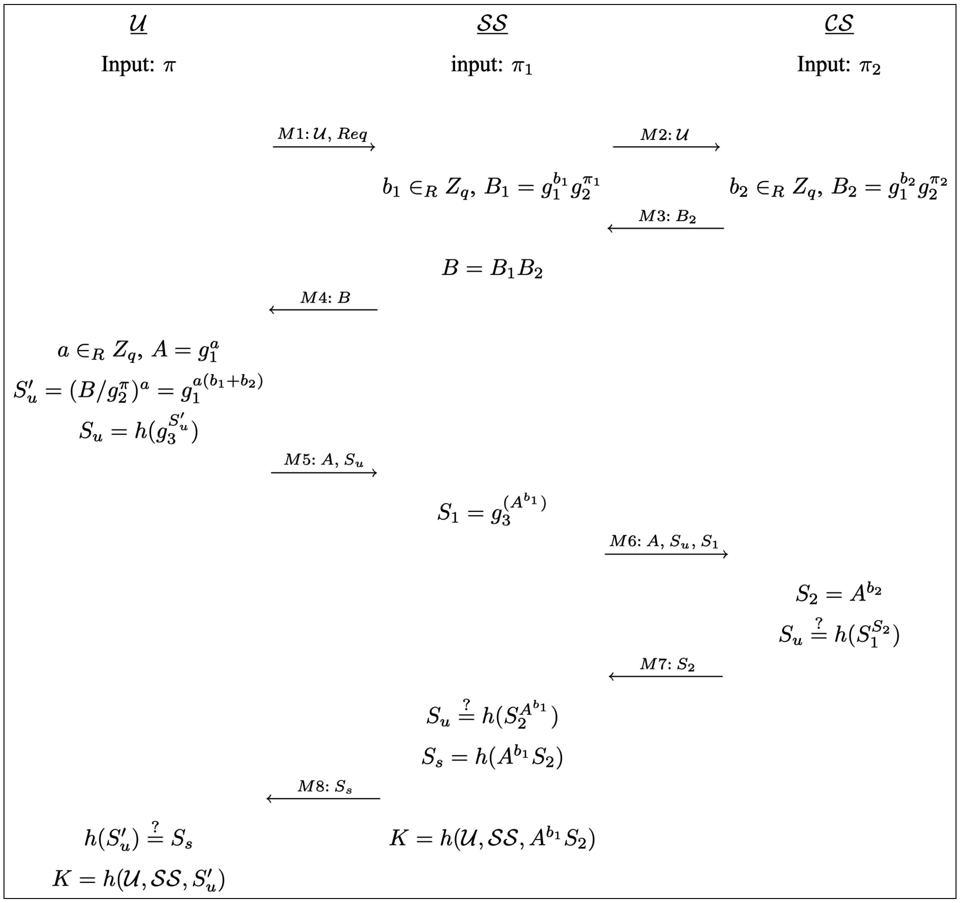
Observe that $\mathcal{CS}$ relies on direct computation of $g_1^{a(b_1+b_2)} (\bmod p)$ to validate the authenticity of $\mathcal{U}$, and the same data is also exploited by $\mathcal{SS}$ and $\mathcal{U}$ to authenticate each other and negotiate a secret session key. This suggests that if $\mathcal{CS}$ were an active attacker, it could establish a session key in the name of $\mathcal{SS}$. This is another reason for $\mathcal{CS}$ being passive.

Claim 2. *The protocol is robust against offline dictionary attacks by $\mathcal{SS}$ as an active adversary.*

Proof. First, if controlled by a passive adversary, help for $\mathcal{SS}$ in terms of offline dictionary attacks is $[A = g_1^a,$ $S_u = h((B/g_2^{\pi})^a)]$ and $[S_2 = A^{b_2},\ B_2 = g_1^{b_2} g_2^{\pi_2}]$. Following a similar analysis as for $\mathcal{CS}$, we can show that $\mathcal{SS}$ is unable to learn anything on either π or π_2 from the two pairs. What remains to consider is when $\mathcal{SS}$ launches active attacks, in which case $\mathcal{SS}$ may behave arbitrarily such as impersonating $\mathcal{U}$ and modifying and replacing messages. From the security analysis for $\mathcal{CS}$, we know that if $\mathcal{SS}$ replaces A coming from $\mathcal{U}$ with g_1^a based on his choice of a and, if this is not detected by $\mathcal{CS}$, $\mathcal{SS}$ can obtain π by offline dictionary attacks. Fortunately, different from the case of $\mathcal{CS}$, this attack cannot succeed for the following reasons: S_2 is sent to $\mathcal{SS}$ in $M7$ only after $\mathcal{CS}$ has already decided on the validity of $S_u \overset{?}{=} h(S_1 A^{b_2})$; it is not possible for $\mathcal{SS}$ to change A and also make $S_u \overset{?}{=} h(S_1 A^{b_2})$ pass the test of $\mathcal{CS}$. As a result, as an active attacker, $\mathcal{SS}$ is still not effective in offline dictionary attacks. □

Claim 3. *The protocol is secure against an active outside adversary controlling no server.*

Proof. Attacks by an active adversary who does not control any server include offline dictionary attacks against user passwords and attempt to acquire the session key K established between $\mathcal{U}$ and $\mathcal{SS}$. For the former, intuitively, such an adversary clearly is no more effective than $\mathcal{SS}$.



$\underline{\mathcal{U}}$
$\underline{\mathcal{SS}}$
$\underline{\mathcal{CS}}$
Input: π
input: π_1
Input: π_2
$M1$: $\mathcal{U}$, Req
$M2$: $\mathcal{U}$
$b_1 \in_R Z_q$, $B_1 = g_1^{b_1} g_2^{\pi_1}$
$b_2 \in_R Z_q$, $B_2 = g_1^{b_2} g_2^{\pi_2}$
$M3$: B_2
$B = B_1 B_2$
$M4$: B
$a \in_R Z_q$, $A = g_1^a$
$S'_u = (B/g_2^{\pi})^a = g_1^{a(b_1+b_2)}$
$S_u = h(g_3^{S'_u})$
$M5$: A, S_u
$S_1 = g_3^{(A^{b_1})}$
$M6$: A, S_u, S_1
$S_2 = A^{b_2}$
$S_u \overset{?}{=} h(S_1^{S_2})$
$M7$: S_2
$S_u \overset{?}{=} h(S_2^{A^{b_1}})$
$S_s = h(A^{b_1} S_2)$
$M8$: S_s
$h(S'_u) \overset{?}{=} S_s$
$K = h(\mathcal{U}, \mathcal{SS}, A^{b_1} S_2)$
$K = h(\mathcal{U}, \mathcal{SS}, S'_u)$

Fig. 3. An improved password authentication and key exchange protocol.

For the latter, the adversary could do as follows: 1) Impersonate any of $\mathcal{U}$, $\mathcal{SS}$, and $\mathcal{CS}$. Clearly, this requires the adversary to derive any of π, π_1, and π_2 by offline dictionary attacks in order for an impersonation to succeed. 2) Compute the value of $g_1^{a(b_1+b_2)} (\bmod p)$ from the protocol transcript. Of help to this end are S_u, S_s, S_1, and S_2. Obviously, inverting S_u and S_s is impossible if the underlying hash function is secure. On the other hand, since the communication channel between $\mathcal{SS}$ and $\mathcal{CS}$ is secret, the attacker cannot observe S_1 and S_2. □

It is interesting to notice that having only one of S_1 and S_2 does not help an outside attacker in computing $g_1^{a(b_1+b_2)} (\bmod p)$. Therefore, one-way secrecy of the channel between $\mathcal{SS}$ and $\mathcal{CS}$ suffices to guarantee the secrecy of the session key.

3.6 An Improved Protocol

There are two weaknesses in the above basic protocol. The first one is made obvious by recalling that we assumed a secret channel between $\mathcal{SS}$ and $\mathcal{CS}$ in the system model. The second one is that $\mathcal{CS}$ can compute the session key established between $\mathcal{U}$ and $\mathcal{SS}$, so $\mathcal{CS}$ gets to know the data exchanged between them. While $\mathcal{CS}$ is passive, this clearly affects the principles of "need to know" and "separation of duty." To address these weaknesses, recall an earlier observation in Section 3.5 that one-way secrecy of the channel between $\mathcal{U}$ and $\mathcal{SS}$ actually suffices in the above basic protocol. Our solution, indeed, takes advantage of this observation by having $\mathcal{SS}$ concealing $A^{b_1} (\bmod p)$ while still enabling $\mathcal{CS}$ for user authentication.

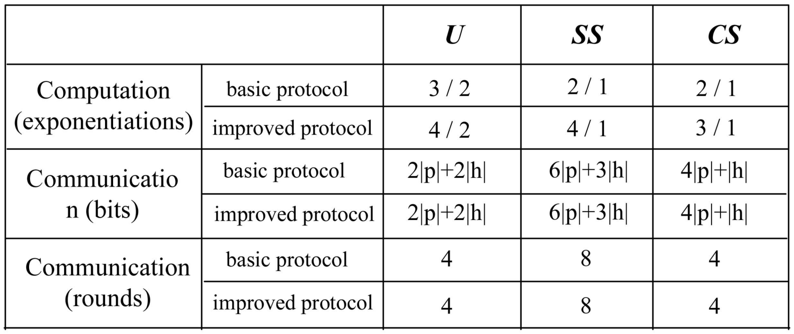
The system setting and the security model are the same as in the basic protocol, except that no secret communication channel between $\mathcal{SS}$ and $\mathcal{CS}$ is assumed. Supposing $\mathcal{U}$ has already registered π_1 to $\mathcal{SS}$ and π_2 to $\mathcal{CS}$ as in the basic protocol, we present an improved password authentication protocol in Fig. 3, where the system parameters are defined in Table 1, arithmetic operations associating with g_1 and g_2 are modulo p, and operations associating with g_3 are modulo Q.

This improved protocol is very similar to the basic protocol in Fig. 2, with the only exception that we introduce the arithmetic operations associating with g_3, which are represented by S_u, S_1 and the checks performed by $\mathcal{SS}$ and $\mathcal{CS}$. By checking it against the basic protocol, we believe it is not hard to understand this improved protocol. So, we do not repeat the process of the protocol execution here. Next, we first check correctness of the protocol.

3.6.1 *Correctness*

For the purpose of verifying $\mathcal{U}$, $\mathcal{CS}$ needs to check $S_u \overset{?}{=} h(S_1^{A^{b_2}} (\bmod Q)$, and $\mathcal{SS}$ needs to check $S_u \overset{?}{=} h(S_1^{S_2} (\bmod Q)$. To make the checks work, it must hold that

$$g_3^{(g_1^{a(b_1+b_2)} \bmod p)} (\bmod Q) = g_3^{(g_1^{ab_1} \bmod p)(g_1^{ab_2} \bmod p)} (\bmod Q).$$



		U	***SS***	***CS***
Computation (exponentiations)	basic protocol	3 / 2	2 / 1	2 / 1
	improved protocol	4 / 2	4 / 1	3 / 1
Communicatio n (bits)	basic protocol	2|p|+2|h|	6|p|+3|h|	4|p|+|h|
	improved protocol	2|p|+2|h|	6|p|+3|h|	4|p|+|h|
Communication (rounds)	basic protocol	4	8	4
	improved protocol	4	8	4

Fig. 4. Performance of the proposed protocols.

Notice that if the order of g_3 is $\bar{p} \neq p$, then the exponentiation parts of g_3 at both sides of the equation are not necessarily equal, that is, $g_1^{a(b_1+b_2)}(\bmod\ p)(\bmod\ \bar{p})$ is of high possibility to not equal $(g_1^{ab_1} \bmod p)(g_1^{ab_2} \bmod p)(\bmod\ \bar{p})$, which, in turn, suggests the equation does not hold. Conversely, if the order of g_3 is p as in the protocol, the equation is bound to hold as

$$g_1^{a(b_1+b_2)}(\bmod\ p)(\bmod\ p) = (g_1^{ab_1} \bmod\ p)(g_1^{ab_2} \bmod\ p)(\bmod\ p).$$

3.6.2 Security

We next examine security of this improved protocol. As we stated, this protocol is quite similar to the basic protocol, except for the introduction of computations associating with g_3. It should be clear that this change makes it no easier to $\mathcal{CS}$ and $\mathcal{SS}$ for the purpose of offline dictionary attacks, as direct computation or guessing of x from $g_3^{g_1^x}(\bmod\ Q)$ is clearly no easier than from $g_1^x(\bmod\ p)$. As a matter of fact, nor does this change make it harder for $\mathcal{CS}$ and $\mathcal{SS}$ with respect to offline dictionary attacks, since it is of equal possibility to guess x from $g_3^{g_1^x}(\bmod\ Q)$ and from $g_1^x(\bmod\ p)$. We thus focus on the effect of removal of the secret channel between $\mathcal{SS}$ and $\mathcal{CS}$, and whether $\mathcal{CS}$ can compute the session key between $\mathcal{U}$ and $\mathcal{SS}$. Clearly, the removal of the secret channel would, in principle, facilitate outside adversaries who do not control any server to derive the session key between $\mathcal{U}$ and $\mathcal{SS}$.

Compared to the basic protocol, an outside adversary additionally gleans $S_1 = g_3^{(A^{b_1})}(\bmod\ Q)$ and $S_2 = A^{b_2}(\bmod\ p)$. The adversary needs to know $A^{b_1}(\bmod\ p)$ in order to derive the session key. However, the additional $S_1 = g_3^{(A^{b_1})}(\bmod\ Q)$ does not help the adversary in computing $A^{b_1}(\bmod\ p)$, which is equivalent to computing the discrete log of S_1. This suggests the removal of the secret channel between $\mathcal{SS}$ and $\mathcal{CS}$ does not, in fact, facilitate the outside adversary. For exactly the same reason, $\mathcal{CS}$ cannot compute the session key either with the knowledge of S_1.

As a result, we have managed to remove the weaknesses contained in the basic protocol.

3.7 Performance of the Protocols

In this section, we examine performance of our proposed two protocols. Let $|p|$ and $|h|$ denote the bit length of p and the hash function $h(.)$, respectively. We outline the performance results in Fig. 4. We have three aspects to evaluate:

1. *Computation performance.* Since exponentiations dominate each party's computation overhead, we only count the number of exponentiations as the computation performance. The digits before "/" denote the total number of exponentiations performed by each party, and the digits following "/" denote the number of exponentiations that can be computed offline. Note that by leveraging on the techniques in [9], each of $g_1^{b_1} g_2^{\pi_1}(\bmod\ p)$ and $g_1^{b_2} g_2^{\pi_2}(\bmod\ p)$ can be computed by a single exponentiation.
2. *Communication performance in terms of bits.* As $|Q|$ is only 1 bit longer than $|p|$, we do not distinguish between $|p|$ and $|Q|$ for ease of comparison. In addition, we have neglected including the bandwidth of $M1$ and $M2$ in this aspect of calculation.
3. *Communication performance in terms of rounds.* One round is a one-way transmission of messages.

Fig. 4 shows that the proposed two protocols demonstrate similar performance and are, in general, quite efficient in terms of both computation and communication to all parties. Take $\mathcal{U}$, for example; it needs to calculate 3 and 4 exponentiations in the two protocols, respectively, and 2 of them can be performed offline. This means $\mathcal{U}$ only computes 1 and 2 exponentiations in real time in the respective protocols; the communication overhead for $\mathcal{U}$ is particularly low in terms of both bits and rounds. As a result, our protocols can readily support wireless applications.

4 Applications

Clearly, our system can be straightforwardly adapted to strengthen existing single-server password systems such as FTP and e-mail systems by adding an additional control server. In such applications, the control server and the service server are most probably managed in the same administrative domain. The system architecture is the same as in Fig. 1d.

We next generalize the basic two-server model to an architecture where a single control server supporting multiple service servers depicted in Fig. 5. In such an architecture, the control server and the service servers are managed in different administrative domains, and the domain where the control server resides enforces more stringent security measurements. More interesting applications can be envisioned for this generalized architecture. A good example of

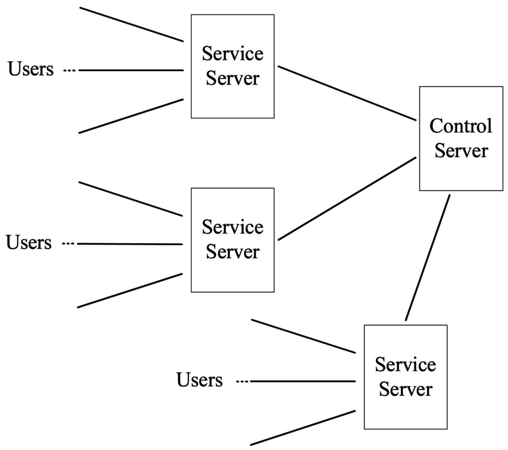


Fig. 5. A generalized two-server architecture of a single control server supporting multiple service servers.

such applications is in a federated enterprise, where many divisions, branches, and affiliations unite under a single enterprise authority. Each of the affiliating organizations that serves different aspect of a business continuum and service coverage has its own business interest and provides service to a distinct group of users. Our proposed system can be implemented in such a federated enterprise as follows: The single control server is managed by the enterprise headquarter that has more funds and better security expertise, while each affiliating organization operates a service server that provides a certain service to its own users.

The generalized two-server architecture and the applications appear to further justify the security model we have assumed upon the earlier two protocols, that is, the control server is controlled by a passive adversary while the service server is controlled by an active adversary. In an enterprise environment, compared to the affiliating organizations, the enterprise headquarter clearly presumably has more budget and better security expertise and, thereby, is in a better position to manage a more trustworthy control server. The competence of the enterprise headquarter should provide a better safeguard not only against outside adversaries but also against inside attackers such as the system administrator. For example, the enterprise headquarter deploys a dedicated control server to restrict insider access, so that only the system administrator has access to the control server.

It is also interesting to note that our proposed protocols technically support the generalized architecture since, from the performance results in Fig. 4, the workload in both computation and communication upon the control server is quite low. Of course, with adequate funds, the headquarter can always deploy a more powerful hardware for the control server or even deploy multiple control servers.

5 Discussions

Our proposed two-server password system together with its practical applications offers many appealing features:

1. A single point of vulnerability, as in the existing password systems, is totally eliminated. Without compromising both servers, no attacker can find user passwords through offline dictionary attacks. The control server being isolated from the public, the chance for it being attacked is substantially minimized, thereby increasing the security of the overall system. As we have shown in the security analysis, the system is also resilient to offline dictionary attacks by outside attackers. This feature allows users to use easy to remember passwords and still have strong authentication and key exchange.
2. The system has no compatibility problem with the single-server model. This is of importance, as most of the existing password systems use a single server.
3. In the system, a password is split into two random numbers. Therefore, a user can use the same password to register to different service servers; they connect either to distinct control servers or to the same control server. This is a highly desirable feature since it makes the system user friendly. A big inconvenience in the traditional password systems is that a user has to memorize different passwords for different applications.
4. The generalization as well as the applications of the two-server password system well support the underlying security model, in the sense that the enterprise headquarter naturally assumes adequate funds and strong security expertise and, therefore, affords and is capable of maintaining a highly trustworthy control server against both inside attackers and outside attackers. Without the concern of a single point of vulnerability, affiliating organizations that operate service servers are offloaded to some extent from strict security management, so they can dedicate their limited expertise and resources to their core competencies and to enhancing service provision to the users.
5. From the perspective of users, they are able to assume the higher creditability of the enterprise while engaging in business with individual affiliating organizations.

It is clear that we have involved the headquarters of a federated enterprise in the partial trust management of its affiliating organizations. One may ask why we do not simply rely on the control server for full trust management in federated enterprise applications, a paradigm similar to Kerberos [18]. First, in practice, each affiliating organization has its own business interest; hence, it has a stake in being involve in the trust management of its own; second and more important, one of the main objectives of our system is to eliminate a single point of vulnerability. In practice, adversaries take on a variety of forms and no security measures and precautions can guarantee that a system will never be penetrated. By avoiding a single point of vulnerability, it gives a system more time to react to attacks.

6 Conclusions and Future Work

In this paper, we proposed a password-based authentication and key exchange system that is built upon a novel two-server model, where only one server communicates to users while the other server stays transparent to the public. Compared with previous solutions, our system possesses many advantages, such as the elimination of a single point of vulnerability, avoidance of PKI, and high efficiency.

In contrast to existing multiserver password systems, our system has great potential for practical applications. It can be directly applied to fortify existing standard single-server password applications, e.g., FTP and Web applications. It can also be applied in the federated enterprise setting, where a single control server supports multiple service servers.

The security model underlying our proposed protocols assumes that the control server can only be controlled by a passive adversary. As we have claimed, this assumption, while strong, is quite logical considering the positioning of the two servers in the two-server model and the applications of the model to federated enterprises. It is, however, clear that weakening of this assumption should be of both practical and theoretical significance, which we shall take as our future work. Our future direction also includes a formal treatment of our proposed system.

Acknowledgments

The authors thank the reviewers for their valuable comments and suggestions, which helped improve the quality of the paper. This work is funded by the Office of Research, Singapore Management University.

References

[1] E. Bresson, O. Chevassut, and D. Pointcheval, "Security Proofs for an Efficient Password-Based Key Exchange," *Proc. ACM Conf. Computer and Comm. Security*, pp. 241-250, 2003.

[2] D. Boneh, "The Decision Diffie-Hellman Problem," *Proc. Third Int'l Algorithmic Number Theory Symp.*, pp. 48-63, 1998.

[3] J. Brainard, A. Juels, B. Kaliski, and M. Szydlo, "A New Two-Server Approach for Authentication with Short Secrets," *Proc. USENIX Security Symp.*, 2003.

[4] S. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks," *Proc. IEEE Symp. Research in Security and Privacy*, pp. 72-84, 1992.

[5] S. Bellovin and M. Merritt, "Augmented Encrypted Key Exchange: A Password-Based Protocol Secure against Dictionary Attacks and Password File Compromise," *Proc. ACM Conf. Computer and Comm. Security*, pp. 244-250, 1993.

[6] M.K. Boyarsky, "Public-Key Cryptography and Password Protocols: The Multi-User Case," *Proc. ACM Conf. Computer and Comm. Security*, pp. 63-72, 1999.

[7] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated Key Exchange Secure Against Dictionary Attacks," *Advances in Cryptology (Eurocrypt '00)*, pp. 139-155, 2000.

[8] M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," *Proc. ACM Computer and Comm. Security*, pp. 62-73, 1993.

[9] V.S. Dimitrov, G.A. Jullien, and W.C. Miller, "Complexity and Fast Algorithms for Multi-Exponentiations," *IEEE Trans. Computers*, vol. 49, no. 2, pp. 141-147, 2000.

[10] W. Ford and B.S. Kaliski Jr., "Server-Assisted Generation of a Strong Secret from a Password," *Proc. IEEE Ninth Int'l Workshop Enabling Technologies*, 2000.

[11] L. Gong, M. Lomas, R. Needham, and J. Saltzer, "Protecting Poorly Chosen Secrets from Guessing Attacks," *IEEE J. Selected Areas in Comm.*, vol. 11, no. 5, pp. 648-656, 1993.

[12] O. Goldreich, *Secure Multi-Party Computation*, working draft, version 1.3, June 2001.

[13] S. Halevi and H. Krawczyk, "Public-Key Cryptography and Password Protocols," *Proc. ACM. Conf. Computer and Comm. Security*, pp. 122-131, 1998.

[14] D.P. Jablon, "Password Authentication Using Multiple Servers," *RSA Security Conf.*, pp. 344-360, 2001.

[15] A. Jain, R. Bolle, and S. Pankanti, *BIOMETRICS: Personal Identification in Networked Society*. Kluwer Academic, 1999.

[16] D.V. Klein, "Foiling the Cracker—A Survey of, and Improvements to, Password Security," *Proc. Second USENIX Security Symp.*, pp. 5-14, 1990.

[17] J. Katz, P.D. Mackenzie, G. Taban, and V.D. Gligor, "Two Server Password-Only Authentication Key Exchange," *Applied Cryptography and Network Security*, pp. 1-16, 2005.

[18] J. Kohl and C. Neuman, *Request for Comments 1510: The Kerberos Network Authentication Service*, Internet Eng. Task Force Network Working Group, 1993.

[19] J. Katz, R. Ostrovsky, and M. Yung, "Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords," *Proc. Advances in Cryptology (Eurocrypt '01)*, pp. 475-494, 2001.

[20] J. Katz, R. Ostrovsky, and M. Yung, "Forward Secrecy in Password-Only Key Exchange Protocols," *Proc. Security in Comm. Networks*, 2002.

[21] P. Mackenzie, T. Shrimpton, and M. Jakobsson, "Threshold Password-Authenticated Key Exchange," *Proc. Advances in Cryptology (Eurocrypt '02)*, pp. 385-400, 2002.

[22] M.D. Raimondo and R. Gennaro, "Provably Secure Threshold Password-Authenticated Key Exchange," *Proc. Advances in Cryptology (Eurocrypt '03)*, pp. 507-523, 2003.

[23] J.M. Wiliams, "Biometrics or ... Biohazards?" *Proc. ACM New Security Pradigms Workshop*, pp. 97-107, 2002.

[24] Y.J. Yang, F. Bao, and R.H. Deng, "A New Architecture for Authentication and Key Exchange Using Password for Federated Enterprises," *Proc. 20th Int'l Federation for Information Processing Int'l Information Security Conf. (SEC '05)*, 2005.

Yanjiang Yang received the BEng and MEng degrees in computer science and engineering from Nanjing University of Aeronautics and Astronautics, China, in 1995 and 1998, respectively, and the MSc degree in biomedical imaging from the National University of Singapore in 2001. He is now a PhD candidate in the School of Computing, National University of Singapore, attached to the Institute for Infocomm Research, A*STAR, Singapore. His research areas include information security and biomedical imaging.

Robert H. Deng received the BEng degree from the National University of Defense Technology, China, in 1978, and the MSc and PhD degrees from the Illinois Institute of Technology in 1983 and 1985, respectively. He is a professor at the School of Information Systems, Singapore Management University. He has more than 140 publications in the areas of error-control coding, digital communications, computer networks, and information security. He served on many advisory and program committees of international conferences. He served as program chair of the 2002 International Conference on Information and Communications Security and the general chair of the 2004 International Workshop on Practice and Theory in Public Key Cryptography. He is a senior member of the IEEE.

Feng Bao received the BSc degree in mathematics, the MSc degree in computer science from Peking University, China, and the PhD degree in computer science from Gunma University, Japan, in 1984, 1986, and 1996, respectively. He was an assistant/associate professor at the Institute of Software, Chinese Academy of Sciences, from 1987 to 1993 and a visiting scholar at Hamberg University, Germany, from 1990 to 1991. Since 1996, he has been with the Institute for Infocomm Research, Singapore. Currently, he is a lead scientist and the head of the Infocomm Security Department and Cryptography Lab at the institute. His research areas include algorithm, automata theory, complexity, cryptography, distributed computing, fault tolerance, and information security. He has over 100 publications and 16 patents.