

3-2006

Fortifying Password Authentication in Integrated Healthcare Delivery Systems

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/10.1145/1128817.1128855>

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the [Information Security Commons](#), and the [Medicine and Health Sciences Commons](#)

Citation

YANG, Yanjiang; DENG, Robert H.; and BAO, Feng. Fortifying Password Authentication in Integrated Healthcare Delivery Systems. (2006). *ASIACCS '06: Proceedings of the ACM Symposium on Information, Computer and Communications Security: Taipei, Taiwan, 21-24 March*. 255-265. Research Collection School Of Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/545

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

Fortifying Password Authentication in Integrated Healthcare Delivery Systems

Yanjiang Yang
School of Information Systems
Singapore Management
University
Singapore 178902
yjyang@smu.edu.sg

Robert H. Deng
School of Information Systems
Singapore Management
University
Singapore 178902
robertdeng@smu.edu.sg

Feng Bao
Institute for Infocomm
Research
Singapore 119613
baofeng@i2r.a-
star.edu.sg

ABSTRACT

Integrated Delivery Systems (IDSs) now become a primary means of care provision in healthcare domain. However, existing password systems (under either the single-server model or the multi-server model) do not provide adequate security when applied to IDSs. We are thus motivated to present a *practical* password authentication system built upon a novel two-server model. We generalize the two-server model to an architecture of a single *control server* supporting multiple *service servers*, tailored to the organizational structure of IDSs. The underlying user authentication and key exchange protocols we propose are password-only, neat, efficient, and robust against off-line dictionary attacks mounted by both servers.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous

General Terms

Security

Keywords

integrated delivery systems (IDSs), password system, user authentication and key exchange, dictionary attack.

1. INTRODUCTION

The application of information technology to health care has driven significant structural changes of the healthcare industry and its methods of care. Taking U.S. for instance, Integrated Delivery Systems (IDSs) are rapidly becoming the primary means of care provision [11]. While their forms vary and will continue to evolve, IDSs generally consolidate under one corporate umbrella multiple types of care providers

(such as hospitals and primary care clinics) that serve different aspects of the care continuum. Some IDSs also include a health care financing arm that offers health plans and pays for care. IDSs depend on integrated information systems to achieve their objectives of cost savings, expansions of market share, and improved quality of care. It is however clear that while the affiliating care providers in an IDS cooperate with one another, each of them has its own business interest, providing distinct services.

Since the advent of computers, entry of a user ID followed by a password has been the most commonly employed means of user identification and authentication. In a password authentication system, each user shares a password or some simple password verification data (PVD) derived from the password with a server, and the user only needs to memorize the password and uses it in user authentication process. One appealing advantage of using password lies in that it has little or no actual cost since no associated physical accessories such as smart cards, sensors, scanners are required. Password authentication still demonstrates vitality in health care, especially as wireless healthcare applications are becoming increasingly prevalent. We are thus concerned with applying password authentication in healthcare information systems, especially in IDSs.

However, password authentication has intrinsic weaknesses, due primarily to the small space of dictionary where passwords are chosen. More specifically, passwords are subject to brute-force *dictionary attacks* where an attacker enumerates every possible password in the dictionary to determine the actual password. The dictionary attacks can occur *on-line* or *off-line*: in an on-line attack, the attacker repeatedly picks a password from the dictionary and login with it to the server by impersonating a legitimate user. Every reject convinces the attacker to eliminate a guess; in an off-line dictionary attack, the attacker records a past successful login between a user and the server, and then checks all the passwords in the dictionary against the gleaned login transcript, until eventually find the correct password. On-line dictionary attacks can be easily thwarted at the system level by limiting the number of unsuccessful login attempts made by a user. In contrast, off-line dictionary attacks are notoriously harder to deal with. As a result, tremendous effort has been dedicated to countering against off-line dictionary attacks in password systems (e.g., [2, 5, 6, 20, 22, 18, 7, 23, 9]). Most of the existing password systems involve a single server, assuming the server is completely reliable. Conse-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ASIACCS'06, March 21-24, 2006, Taipei, Taiwan.
Copyright 2006 ACM 1-59593-272-0/06/0003 ...\$5.00.

quently, these systems are precisely robust to off-line dictionary attacks by outside attackers. As a matter of fact, the server in these systems can get the password directly or from the PVD by off-line dictionary attacks.

Unfortunately, a fully trusted server is hard to achieve in practice, due to either outsider penetrations or unscrupulous insiders such as system administrators. As a result, deploying single-server password systems in IDSs, that is, each affiliating care provider in an IDS maintains by itself a server housing passwords of its users, is not an optimal practice. This is because (1) some affiliating care providers may lack sufficient expertise and funds to sustain servers robust enough against outsider penetrations and insider attacks; (2) it is now a trend that some organizations choose to outsource their IT management to external specialized service providers so as to reduce administrative costs. In such cases, system administrators may present themselves as a big threat to system security [8]. Solutions to the problem of unreliable server is to distribute passwords as well as the verification functionality to *multiple* servers, so that an attacker is forced to compromise several servers for the purpose of off-line dictionary attacks (e.g. [14, 19, 24, 25]). Clearly, these systems based on multiple servers do not directly apply to IDSs either, since we cannot expect every affiliating care provider to afford deploying multiple servers. We shall further discuss the disadvantages of multi-server password systems in Section 3.

Considering the organizational structure of IDSs, it is a natural solution that the corporate authority of an IDS gets involved into the trust management of its affiliating care providers. Following this rationale, we present a *practical* password authentication system built upon a novel two-server model: each affiliating care provider operates a front-end *service server*, providing services to its users; the corporate authority manages a back-end *control server*, whose sole purpose is to assist the service servers in user authentication. The two-server system follows the principle of multiple servers by dispensing passwords and verification functionality to two servers, but distinguishing from other multi-server systems in that only the service server exposes to users while the control server stays transparent to the public. Our system requires no PKI, which is of particularly advantage considering PKIs are proven notoriously expensive in real world deployment. While we discuss the two-server system in the context of IDSs, it clearly has wider applicability.

The rest of the paper is organized as follows. In Section 2, we review related work. In Section 3, we discuss different server models for password systems and generalize the two-server model to the context of IDSs. We then give a basic two-server password authentication and key exchange protocol in Section 4. In Section 5, we improve the basic protocol by circumventing the weaknesses contained in it, and in Section 6, we present the third protocol. We give some discussions on the protocols in Section 7, and Section 8 contains the concluding remarks and future work.

2. 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 off-line dictionary attacks, whereas the involvement of *public key cryptosystems* (e. g., public key encryption and digital signature schemes) is not essential [18]. This observation dif-

ferentiates two separate approaches to the development of secure password systems: combined use of password and public key cryptosystem, and 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 authentication systems include [15, 18, 7]. In these systems, the use of public key cryptosystems 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, known as password-only authenticated key exchange or PAKE, have been extensively studied (e.g., [2, 5, 6, 22, 23, 9]). The PAKE protocols do not involve any public key cryptosystem and therefore are much more attractive for real world applications. It is our belief that any use of public key cryptosystems in a password 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 PVD with a single authentication server. While these systems are sufficiently robust against off-line dictionary attacks mounted by outsiders, they are by no means resilient to attacks initiated at the server side, e.g., in the event of server break-ins by outsiders or misbehavior by unscrupulous system administrators. To address this problem, password systems based on multiple servers were proposed. The principle of such multi-server systems is distributing the password/PVD database as well as the verification functionality to multiple servers in order to eliminate the single point of vulnerability. As such, without compromising multiple servers, an attacker is bound not to be effective in off-line dictionary attacks. The idea of splitting a secret into multiple segments has also been leveraged by several systems other than the multi-server password systems for other purposes. For example, it was suggested to split a private RSA signing key into several parts in order to enable “mutlsignatures” [1] or to mitigate the catastrophic consequences of a Kerberos server that houses the secret keys of all its users being compromised [17]. The system in [14], believed to be the first multi-server password system, splits a short password among multiple servers who then collaborate on user authentication. However, the servers in [14] need to use public keys. An improved version of [14] was proposed in [19] which eliminates the use of public keys while achieving similar security. Further and more rigorous extensions were due to [24] and [25], where the former built a *t*-out-of-*n* threshold PAKE protocol and provided a formal security proof under the random oracle model [10], and the latter presented two provably secure threshold PAKE protocols under the standard model. While the protocols in [24] and [25] are theoretically significant, they have low efficiency and high operational overhead. In these multi-server 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 (see Section 3).

The password system most closely related to ours is the two-server system recently proposed by Brainard *et al.* [4], where one server exposes itself to users and the other is hidden from the public. While this two-server setting is ef-

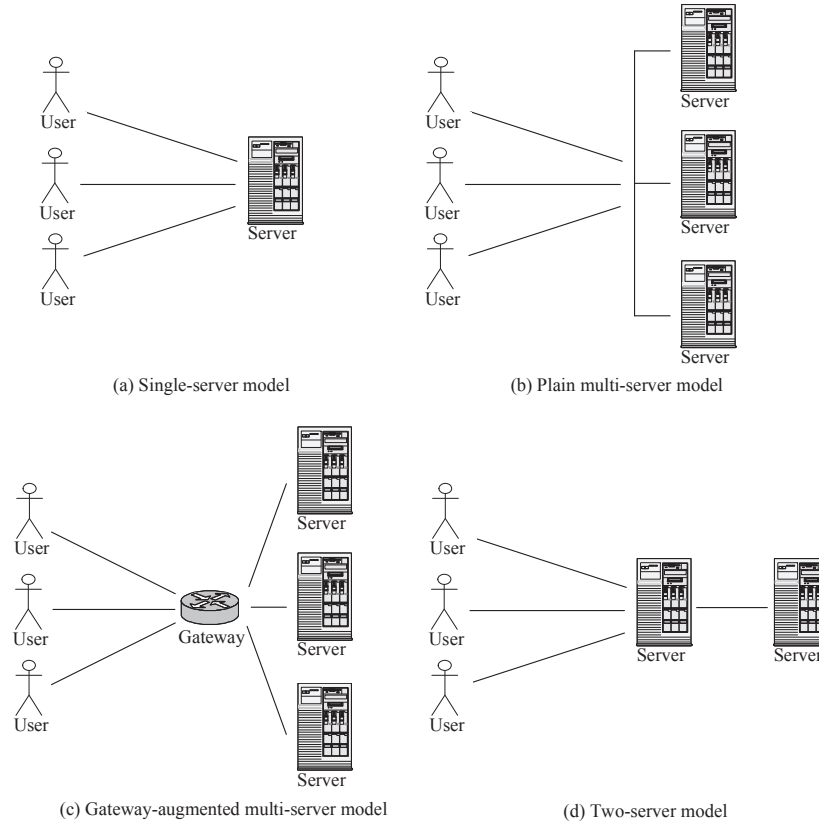


Figure 1: Models of password systems.

efficient, it is not a password-only system: both servers need to have public keys to protect the communication channels between users and servers. As we have stressed earlier, this makes it difficult to fully enjoy the benefits of a password system. In addition, the system in [4] 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.* [26] 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 [26] is that only the back-end server holds a public key. Nevertheless, the system in [26] is still not a password-only system. We notice that the two-server system presented in [21] does not follow the two-server model in [4, 26], but is a special case of multi-server systems. This should be clear shortly in Section 3. Our work in this paper generalize the two-server model in [4, 26] to the context of IDSs, and we adopt a very different method in the protocol design. As a result, our protocols require no public key cryptosystem whatsoever, and are quite efficient and neat.

3. A TWO-SERVER MODEL AND ITS APPLICATION TO IDS

Password systems in the literature are generally built over the following four types of architectures shown in Figure 1. The first type is the *single-server model* given in Figure 1(a), where a single server is leveraged 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 with respect to off-line dictionary attacks against the user password database.

The second type is the *plain multi-server model* depicted in Figure 1(b), in which the server side comprises multiple servers for the purpose of removal of 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 for authentication. Clearly, the main problem with the plain multi-server 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 [14, 19, 24] and one of the two protocols in [25] assume this model.

The third type is the *gateway-augmented multi-server model* outlined in Figure 1(c), 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 multi-server model. However, the gateway introduces an additional layer in the architecture, which appears “redundant” since the sole role of the gateway is to relay messages between users and servers, and it does not in any way involve in service provision, user authentication and other security enforcements. From security perspective, more components

in general imply more points of vulnerabilities. Protocols based on the gateway-augmented multi-server model include [25] and [21].

The fourth type is the *two-server model* outlined in Figure 1(d), that comprises two servers at the server side, one of which is a *public server* exposing itself to users and the other is a *back-end server* staying behind the scene; users only contact the public server, but the two servers cooperate to authenticate users. It is important to note the essential differences between the two-server model and the earlier multi-server 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 multi-server models, a user establishes a session key (either different or same) with each of the servers. For exactly this reason, we view the two-server system in [21] as a special case of the gateway-augmented multi-server model of two servers; (2) from security point of view, servers in the multi-server models are equally exposed to outside attackers (recall that the gateway in the gateway-augmented multi-server 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.

It is clear that the two-server model has successfully eliminated drawbacks in the plain multi-server model (i.e., simultaneous communications between a user and multiple servers) and the gateway-augmented multi-server model (i.e., redundancy), while allows for distribution of user passwords and the verification functionality to two servers so as to eliminate the 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. We are thus motivated to adapt the two-server model to the context of IDSs. To that end, we specify that 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 enables to enforce clear *separation of duty* in our system. We emphasize that in the plain multi-server model and the gateway-augmented multi-server model, several or all servers equally participate in user authentication as well as service provision (this is implied by the fact that a user negotiates a session key with each server).

Recall that an IDS is formed by many affiliating care providers united under a single corporate authority shown in Figure 2(a), where each affiliating organization serves a different aspect of care provision, has its own business interest, and has a distinct group of users. We generalize the above two-server model to an architecture of a single control server supporting multiple service servers outlined in Figure 2(b). As such, application of this generalized architecture to an IDS is as follows: the corporate authority of the IDS manages the control server, and each affiliating care provider operates a service server that provides a certain care service to its own users.

The two-server model we employ together with its application to IDSs enjoys many advantages:

- A single point of vulnerability as in the single-server model is totally eliminated. In principle, without compromising both servers no attacker can find user pass-

words through off-line dictionary attacks. On the other hand, as the control server is isolated from the public, the chance for it being attacked is substantially minimized, thereby increasing server side security and in turn security of the overall system.

- It decreases the demand of bandwidth as well as synchronization at the user side since users do not engage in simultaneous communication with multiple servers. This is of particular importance when supporting wireless healthcare applications.
- In an IDS, the corporate authority naturally assumes strong security expertise and sufficient funds, and is thus in a much better position to operate and maintain the control server. Without the worry of a single point of vulnerability, the affiliating care providers 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.
- From the perspective of users, they are offered a way to assume the higher creditability of an IDS as a whole, while engaging business with individual affiliating care providers.

It is clear that we have involved the corporate authority of an IDS into the (partial) trust management of its affiliating care providers. One may wonder why we simply delegate full trust management of the affiliating care providers to the corporate authority. First, in practice each affiliating care provider has its own business interest, hence it has a stake to involve into the trust management of its own; second and more importantly, one of the main objectives of our system is to eliminate a single point of vulnerability. Adversaries take on a variety of forms in the real world, and no security measures and precautions can guarantee that a system will never be penetrated.

4. A BASIC TWO-SERVER PASSWORD AUTHENTICATION AND KEY EXCHANGE PROTOCOL

As discussed earlier, the existing password protocols upon the two-server model such as [4, 26] are not password only systems and have weaknesses. In this and the next two sections, we present *practical* password authentication and key exchange protocols following the two-server model and the generalized architecture. In particular, we propose three protocols: the first protocol is based on a security model that assumes the service server is controlled by an active adversary while the control server is controlled by a passive adversary; the second protocol assumes the same security model but circumvents some weaknesses contained in the first protocol; the third protocol strengthens the security model such that both servers are controlled by active adversaries. All protocols are password-only, requiring no public key cryptosystem. We stress that each of these protocols suffices for practical use and is of independent interest. We start by listing the notations that are used in the sequel in Table 1.

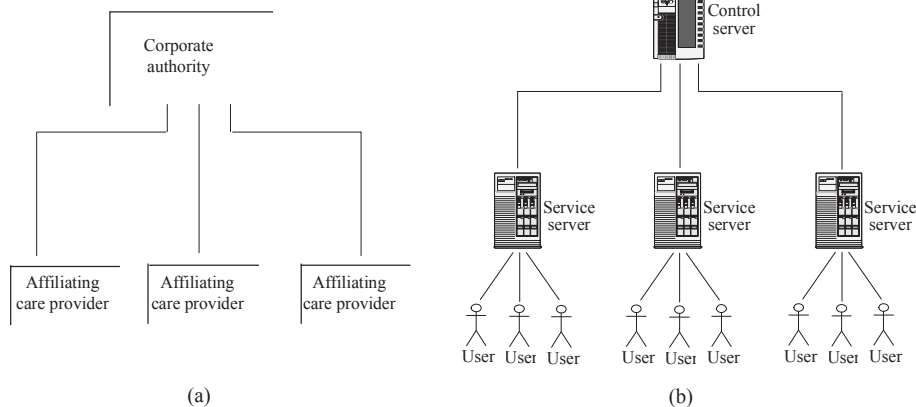


Figure 2: (a) Organizational architecture of IDSs (b) Generalization of the two-server model to IDSs

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.
$h(\cdot)$	A cryptographic hash function modelled as the random oracle [10].
$\mathcal{U}, \mathcal{SS}, \mathcal{CS}$	Identity of user, service sever and control server, respectively.

4.1 System Model

Three types of entities are involved in the protocol, i.e., users, service servers, and a control server. They are organized following the generalized architecture in Figure 2(b), and each pair of a service server and the control server follows the two-server model in Figure 1(d). In this setting, a user \mathcal{U} only communicates with the service server \mathcal{SS} he registered to and does not necessarily know the control server \mathcal{CS} . For the purpose of user authentication and key exchange, \mathcal{U} uses a short 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 for this protocol: \mathcal{CS} is controlled by a *passive adversary* and \mathcal{SS} is controlled by an *active adversary*, in terms of off-line dictionary attacks to user passwords, but they do not collude (otherwise it is equal to the single-server model). By definition (e.g., [16]), a passive adversary follows honest-but-curious behavior, that is, it honestly executes the protocol according to the protocol specifications 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 a passive adversary controls a server, it knows all internal states of and knowledge known to the server including the long term secret key materials (if any) and user password shares. In contrary, an active adversary can act arbitrarily in order to uncover user passwords. In addition, we assume a secret communication channel between \mathcal{SS} and \mathcal{CS} . We shall discuss how to remove this assumption in the subsequent protocols.

Note that in this security model, while the control server acting as a passive adversary is a relatively strong assumption, it is quite *reasonable* considering the positioning of the servers in the two-server model and the application scenario of the model in IDSs. First, the control server is clearly more trustworthy than the service server in the two-server model as we discussed earlier. Second, consider the application scenario of the two-server model in IDSs, where the control server supports multiple service servers and is managed by the corporate authority: on the one hand, the control server deserves more investment for a higher level of security, since it supports multiple service servers; on the other hand, the corporate authority affords investing more upon the control server, since the corporate authority presumably has more funds and security expertise.

4.2 A High Level Description

Central to our protocol design is to counter against off-line 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 each of them is no longer subject to off-line dictionary attacks, and then distribute the shares to the two servers. As a consequence, an attacker cannot succeed in off-line 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 *user 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} who uses π_2 .

4.3 User Registration

In any password system, to enrol as a legitimate user in a service, a user must beforehand register to the service provider by establishing a password with the provider. In our system, \mathcal{U} needs to register to not only the actual service provider \mathcal{SS} but also the control server \mathcal{CS} . Suppose \mathcal{U} has already successfully identified to \mathcal{SS} , e.g., by showing his identity card, \mathcal{U} splits his password π into two long number

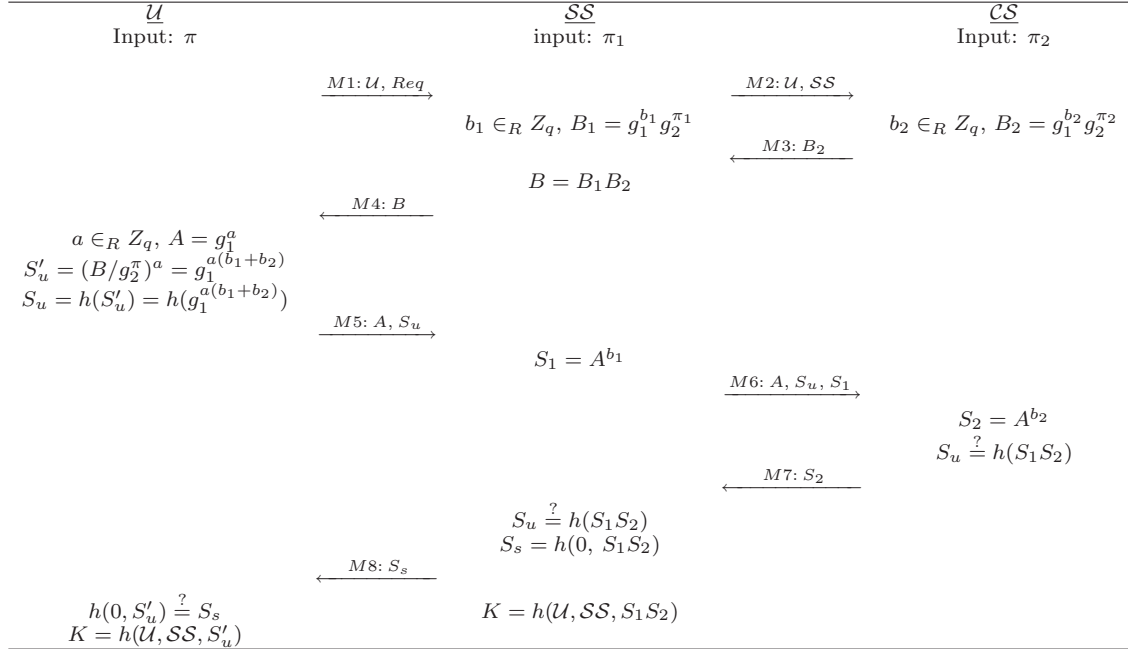


Figure 3: A Basic Password Authentication and Key Exchange Protocol.

$\pi_1 \in_R Z_q^*$ and $\pi_2 \in_R Z_q^*$ such that $\pi_1 + \pi_2 = \pi \pmod{q}$, where q is a prime defined in Table 1. \mathcal{U} then registers in a secure way π_1 and π_2 to \mathcal{SS} and \mathcal{CS} , respectively. \mathcal{SS} stores (\mathcal{U}, π_1) to its secret user account database, and \mathcal{CS} stores $(\mathcal{U}, \pi_2, \mathcal{SS})$ to its secret user account database. A user may register to different service servers, so \mathcal{CS} must bind the user identity \mathcal{U} to the service server \mathcal{SS} it registered to in the user account. This completes the user registration step. Recall however that \mathcal{CS} stays hidden from \mathcal{U} , then one may wonder how \mathcal{U} registers π_2 to \mathcal{CS} . This actually is not a problem in practice: \mathcal{U} can reach \mathcal{CS} (actually the corporate authority that manages \mathcal{CS}) through out-of-band channels, such as postal mail. Indeed, imagine that a user enrolls in a medical clinic, it is not strange at all that the user still needs to submit a secret to a higher authority of the clinic so as to activate his account.

4.4 Protocol

Let p, q, g_1, g_2 and $h(\cdot)$ be defined in Table 1, we outline the basic password authentication protocol in Figure 3, which enables mutual authentication and key exchange between \mathcal{U} and \mathcal{SS} . In the figure, we omitted the modulo p notation for arithmetic operations, as this should be clear from the context.

Let us take a close look at the protocol step by step. 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} 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 $B = B_1 B_2 \pmod{p}$ to \mathcal{U} in $M4$. After receiving $M4$, \mathcal{U} selects $a \in_R Z_q$, and computes $A = g_1^a \pmod{p}$, $S'_u = (B/g_2^\pi)^a = g_1^{a(b_1+b_2)} \pmod{p}$ 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 $S_1 = A^{b_1} \pmod{p}$ and sends S_1, A and S_u to \mathcal{CS} in $M6$. Upon receipt of $M6$, \mathcal{CS} computes $S_2 = A^{b_2} \pmod{p}$ and checks whether $S_u \stackrel{?}{=} h(S_1 S_2) = h(g_1^{a(b_1+b_2)})$: if it holds, \mathcal{CS} is assured of the authenticity of \mathcal{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 $S_u \stackrel{?}{=} h(S_1 S_2)$: 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 $S_s = h(0, S_1 S_2)$, and sends S_s to \mathcal{U} in $M8$ and afterwards computes a session key $K = h(\mathcal{U}, \mathcal{SS}, S_1 S_2)$; otherwise, \mathcal{SS} aborts the protocol. Upon receiving $M8$, \mathcal{U} checks if $h(0, S'_u) \stackrel{?}{=} S_s$: 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.

4.5 Security Analysis

In what follows, we analyze security of the basic protocol. Our analysis is based on the following Decisional Diffie-Hellman (DDH) assumption [3]:

DDH Assumption: let p, q be defined as in Table 1, and $g, h \in_R QR_p$, for every probabilistic polynomial time algorithm \mathcal{A} , the following condition is satisfied:

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

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 off-line 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 off-line 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 off-line dictionary attacks against the password of \mathcal{U} . Clearly \mathcal{CS} can obtain $B_1 = B/B_2 = g_1^{b_1} g_2^{\pi_1} \pmod{p}$ from $M4$. However, from B_1 alone, \mathcal{CS} cannot learn anything on π_1 in an information theoretic sense. What remains relevant to \mathcal{CS} for off-line 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 off-line dictionary attacks, we thus suppose \mathcal{CS} knows S'_u for ease of analysis. Note that $A = g_1^a \Rightarrow g_1 = A^{a^{-1}} \pmod{p}$ and $S'_u = (B/g_2^\pi)^a \Rightarrow B/g_2^\pi = S'^{a^{-1}}_u \pmod{p}$. Under the DDH assumption, \mathcal{CS} cannot distinguish between $[A, g_1 = A^{a^{-1}}, S'_u, B/g_2^\pi = S'^{a^{-1}}_u]$ 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, $B_1 = g_1^{b_1} g_2^{\pi_1} \Rightarrow B/g_2^{\pi_1} = g_1^{b_1} \pmod{p}$, 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 off-line dictionary attacks. This completes the proof. \square

It is important to note that in the above analysis, we have implicitly assumed that \mathcal{CS} does not know $a = \log g_1^A \pmod{q}$. 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 $A = g_1^a \pmod{p}$. \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 off-line dictionary attacks. To see this, consider the second pair where \mathcal{CS} knows $a = \log g_1^A$ 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 $g_1^{a(b_1+b_2)} \pmod{p}$ to validate the authenticity of \mathcal{U} , and \mathcal{SS} and \mathcal{U} use the same data to authenticate each other and negotiate a secret session key. This suggests that if \mathcal{CS} were an active adversary, 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 off-line dictionary attacks by \mathcal{SS} as an active adversary.*

PROOF. First, if controlled by a passive adversary, of help for \mathcal{SS} in terms of off-line 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} , 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 off-line 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 $S_u \stackrel{?}{=} h(S_1 A^{b_2})$; it is not possible for \mathcal{SS} to change A and also make $S_u \stackrel{?}{=} h(S_1 A^{b_2})$ pass the test of \mathcal{CS} . As a result, as an active attacker, \mathcal{SS} is still not effective in off-line dictionary attacks. \square

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

PROOF. Attacks by an active outside adversary who does not control any server include off-line 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} . For the latter, the adversary could do as follows: (1) to 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 off-line dictionary attacks in order for a successful impersonation; (2) computing the value of $g_1^{a(b_1+b_2)} \pmod{p}$ 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 . \square

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

5. AN IMPROVED PROTOCOL

There have two weaknesses in the above basic protocol. The first one is 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} could get to know the data exchanged between them. While \mathcal{CS} is a passive adversary, this clearly affects the “need to know” principle. To address these weaknesses, recall an earlier observation 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 \mathcal{SS} concealing $A^{b_1} \pmod{p}$ from \mathcal{CS} while still enabling \mathcal{CS} to accomplish the task of 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. Suppose \mathcal{U} has already registered π_1 to \mathcal{SS} and π_2 to \mathcal{CS} as in the above protocol, we present an improved password authentication protocol in Figure 4, where the system parameters are defined in Table 1, and arithmetic operations associating with g_1 and g_2 are modulo p , while operations associating with g_3 are modulo Q .

By checking it against the basic protocol in Figure 3, it is not hard to understand this improved protocol. So we do not repeat the process of the protocol execution. Next, we first check correctness of the protocol.

5.1 Correctness

For the purpose of verifying \mathcal{U} , \mathcal{CS} needs to check $S_u \stackrel{?}{=} h(S_1^{A^{b_2}} \pmod{Q})$, and \mathcal{SS} needs to check $S_u \stackrel{?}{=} h(S_1^{S_2} \pmod{Q})$.

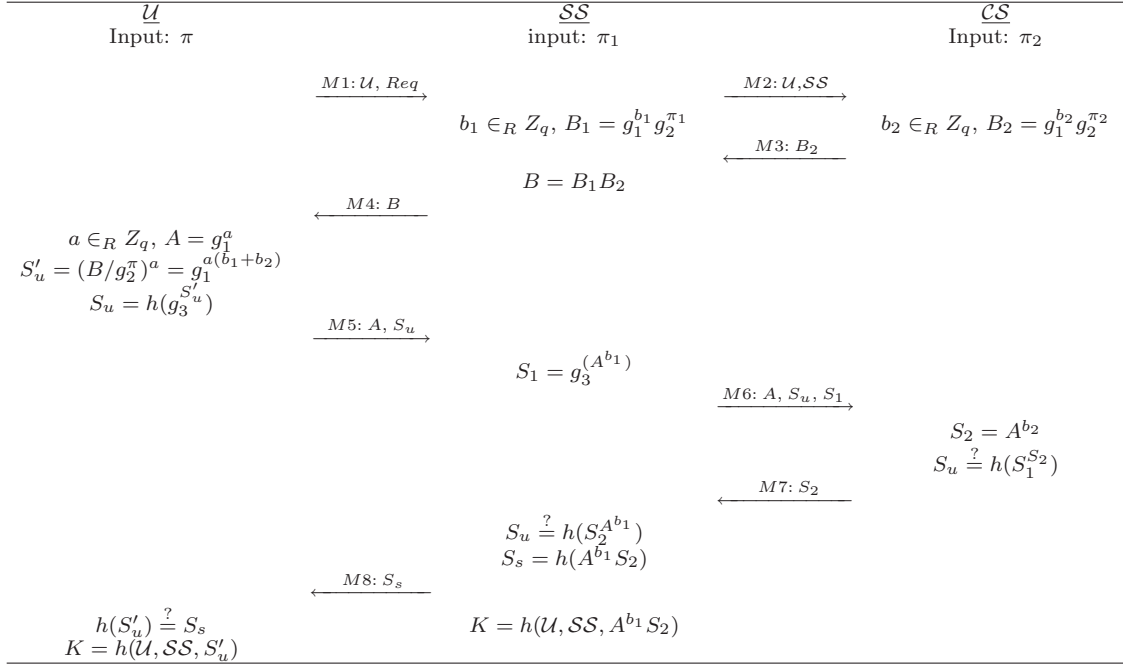


Figure 4: An Improved Password Authentication and Key Exchange Protocol.

To make the checks work, it must hold that $g_3^{(g_1^{a(b_1+b_2)} \bmod p)}$ (mod Q) = $g_3^{(g_1^{ab_1} \bmod p)(g_1^{ab_2} \bmod p)}$ (mod Q). However, normally $g_1^{a(b_1+b_2)} \bmod p \neq (g_1^{ab_1} \bmod p)(g_1^{ab_2} \bmod p)$, but it holds that $g_1^{a(b_1+b_2)} \bmod p = (g_1^{ab_1} \bmod p)(g_1^{ab_2} \bmod p)$ (mod p). As $g_3 \in QR_Q$ is of order p , the above checks thus hold.

5.2 Security

We now examine security of the improved protocol. This protocol is quite similar to the basic protocol, except for the introduction of computations associating with g_3 . It is clear that this change makes it no easier (nor harder in fact) to \mathcal{CS} and \mathcal{SS} for off-line dictionary attacks. 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 for deriving 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 datum $S_1 = g_3^{(A^{b_1})} \bmod Q$ does not further help the adversary compute $A^{b_1} \bmod p$, which is clearly equivalent to computing the discrete logarithm of S_1 . This suggests that the removal of the secret channel between \mathcal{SS} and \mathcal{CS} does not in fact facilitate an outside adversary. For exactly the same reason, \mathcal{CS} cannot compute the session key with the knowledge of S_1 either.

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

6. THE THIRD PASSWORD AUTHENTICATED PROTOCOL

Recall that the security model underlying the earlier two 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 application of the model to IDSs. The earlier two protocols, especially the second one thus suffice when applied to IDSs. It is however clear that weakening of the assumption that the control server can only be controlled by a passive adversary is of both practical and theoretical significance.

Consequently, the security model for the third protocol is that both \mathcal{SS} and \mathcal{CS} are controlled by active adversaries and they do not collude. Moreover, no secret communication channel is assumed. Based on the second protocol in Figure 4, we present the third protocol in Figure 5.

By referring to the second protocol, it is not hard to understand and check the correctness of this protocol. So we do not repeat the protocol execution and the correctness checking of the protocol, and only focus on its security in what follows.

6.1 Security Analysis

A main difference between the second protocol in Figure 4 and this protocol is the additional two messages $M5$ and $M6$ between \mathcal{U} and \mathcal{SS} . We however notice that these messages themselves do not leak more information on the password π . We thus only concern with the case when \mathcal{CS} is controlled by an active adversary. From the security analysis for the earlier two protocols, we know that \mathcal{CS} can be successful in off-line dictionary attacks towards \mathcal{SS} when it knows $a = \log g_1^A \bmod q$ in the earlier two protocols. The vulnerability actually results from the fact that \mathcal{U} itself takes full charge of computing A , which allows \mathcal{CS} to replace A or to impersonate \mathcal{U} against \mathcal{SS} . In this final protocol, the additional two messages $M5$ and $M6$ actually regulate \hat{A}

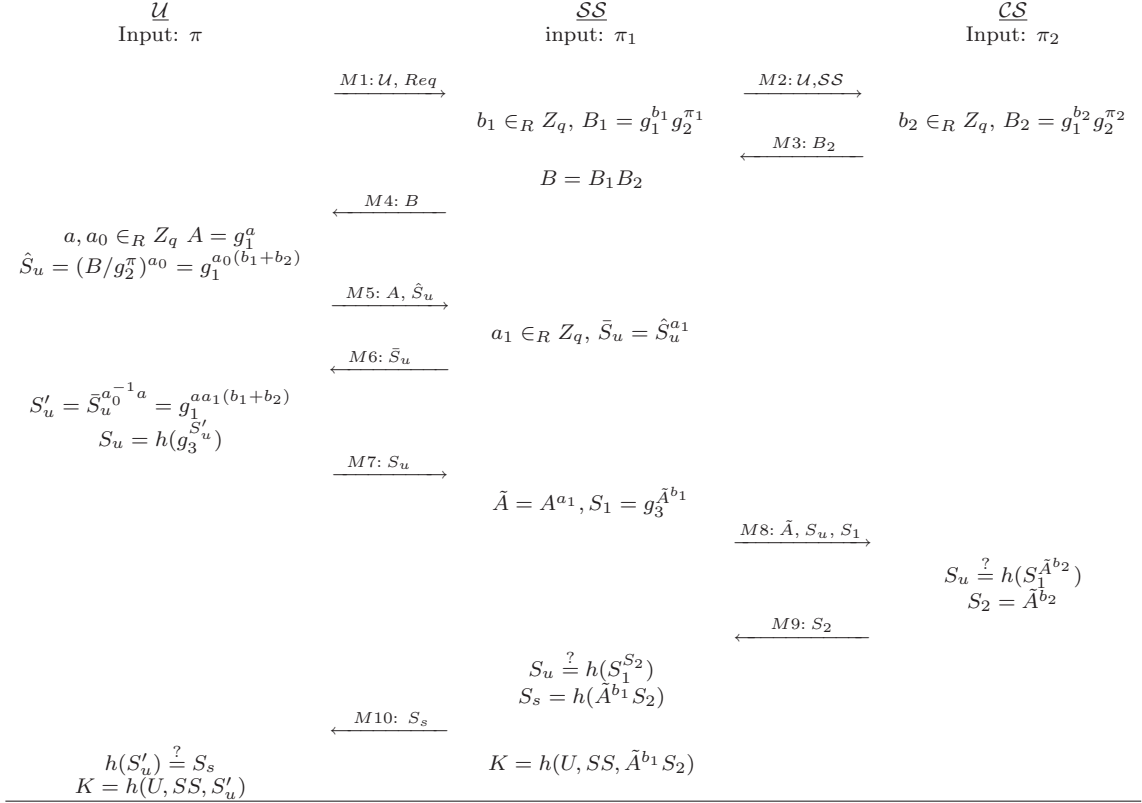


Figure 5: Another Password Authentication and Key Exchange Protocol.

(which is equivalent to A in the earlier two protocols) to be constructed by both \mathcal{U} and \mathcal{SS} . As a result, while \mathcal{CS} , as an active adversary, can manipulate the part due to \mathcal{U} (i.e., a) by replacement or impersonation, it cannot manipulate the part due to \mathcal{SS} (i.e., a_1). That is, \mathcal{CS} cannot compute the discrete logarithm of \tilde{A} to the base g_1 , and thus is not effective in off-line dictionary attacks as analyzed in the earlier two protocols.

A potential weakness of the protocol lies in that an active outside adversary between \mathcal{U} and \mathcal{SS} can influence the session key shared by \mathcal{U} and \mathcal{SS} . To see this, the adversary could choose a random number $\beta \in_R Z_q$ and intercept $M5$; then compute $A = A^\beta \pmod{p}$, $\hat{S}_u = \hat{S}_u^\beta \pmod{p}$, and continue the protocol by sending the manipulated elements A and \hat{S}_u to \mathcal{SS} as in the original protocol; other part of the protocol remains intact. It is easy to check that the protocol still works, but \mathcal{U} and \mathcal{SS} ends up sharing a session key $K = h(U, SS, g_1^{aa_1\beta(b_1+b_2)})$, which is distinct from the one perceived in the original protocol. We however notice that attacks of this kind do not affect the secrecy of the session key and the user password, nor do they affect the authentication functionality of the protocol. The adversary gains no advantages from such attacks. We therefor do not consider them as a serious issue.

7. DISCUSSIONS

In this section, we first examine performance of our proposed protocols. Let $|p|$ and $|h|$ denote the bit length of p and the hash function $h(\cdot)$, respectively, we outline the performance results in Figure 6. We have three aspects to

evaluate. (1) *Computation performance*: as exponentiation operations dominate each party's overhead, we only count the number of exponentiations as the computation performance, and the digits following / denote the number of exponentiations that can be computed off-line. Note that by leveraging on the techniques in [12], each of $g_1^{b_1} g_2^{\pi_1} \pmod{p}$ and $g_1^{b_2} g_2^{\pi_2} \pmod{p}$ can be computed by a single exponentiation. (2) *Communication performance in terms of bits*: as $|Q|$ is only 1 bits longer than $|p|$, we do not explicitly 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 represents a one way transmission of messages.

		\mathcal{U}	\mathcal{SS}	\mathcal{CS}
Computation (exponentiations)	1st protocol	3 / 2	2 / 1	2 / 1
	2nd protocol	4 / 2	4 / 1	3 / 1
	3rd protocol	5 / 2	6 / 1	3 / 1
Communication (bits)	1st protocol	$2 p +2 h $	$6 p +3 h $	$4 p + h $
	2nd protocol	$2 p +2 h $	$6 p +3 h $	$4 p + h $
	3rd protocol	$4 p +2 h $	$8 p +3 h $	$4 p + h $
Communication (rounds)	1st protocol	4	8	4
	2nd protocol	4	8	4
	3rd protocol	6	10	4

Figure 6: Performance of the proposed protocols.

From the table, it shows that the proposed protocols 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, 4 and 5 exponentiations in the three protocols, respectively, and 2 of them can be performed off-line in all cases; the communication overhead for \mathcal{U} is also quite low in terms of both bits and rounds. As a result, our protocols can readily apply to wireless applications. It is also worth noticing from the table that our proposed protocols favor the generalized architecture for IDSs where one control server supports multiple service servers, since the workload in terms of both computation and communication upon the control server is particularly low. Of course, with sufficient funds the corporate authority of an IDS can always deploy a powerful server.

Finally, it is interesting to note that in our protocols, a password is splitted into two random shares; therefore, a user can use the same password to register to different service servers, either they connect 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.

8. CONCLUSIONS AND FUTURE WORK

Use of password is still a vital means of user authentication in healthcare information systems, especially as wireless healthcare applications are becoming increasingly prevalent. However, existing password systems have difficulties when applied to IDSs, the current primary means of care provision. To solve this problem, we presented a practical password authentication system based on a novel two-server model, and generalized the model to an architecture of a single control server supporting multiple service servers by considering the organizational structure of IDSs. The underlying user authentication and key exchange protocols we proposed are password-only, neat, efficient, and robust against off-line dictionary attacks mounted by both servers.

While we examined the security of the proposed protocols, a formal treatment of the system is necessary. One of our future work is thus to formally define and validate the security of the system. Another direction of the future work consists of investigating other issues associated with password authentication such as phishing attacks [13].

9. ACKNOWLEDGMENTS

We thank the anonymous referees for their helpful comments and suggestion.

This work is funded by Office of Research, Singapore Management University.

10. REFERENCES

- [1] C. Boyd. Digital Multisignatures, *Cryptography and Coding*, pp 241-246, 1989.
- [2] E. Bresson, O. Chevassut, D. Pointcheval. Security Proofs for an Efficient Password-Based Key Exchange, *ACM. Computer and Communication Security*, pp. 241-250, 2003.
- [3] D. Boneh. The Decision Diffie-Hellman Problem, *3rd International Algorithmic Number Theory Symposium*, LNCS 1423, pp. 48-63, 1998.
- [4] J. Brainard, A. Juels, B. Kaliski, M. Szydlo. A New Two-Server Approach for Authentication with Short Secret, *Proc. USENIX Security*, 2003.
- [5] S. Bellovin, and M. Merritt. Encrypted Key Exchange: Password- Based Protocols Secure Against Dictionary Attacks, *IEEE Symposium on Research in Security and Privacy*, pp. 72-84, 1992.
- [6] S. Bellovin, and M. Merritt. Augmented Encrypted Key Exchange: A Password-Based Protocol Secure Against Dictionary Attacks and Password File Compromise, *ACM. Computer and Communication Security*, pp. 244-250, 1993.
- [7] M. K. Boyarsky. Public-key Cryptography and Password Protocols: The Multi-User Case, *ACM Conference on Computer and Communication Security*, pp. 63-72, 1999.
- [8] L. Bouganim, P. Pucheral. Chip-Secured Data Access: Confidential Data on Untrusted Servers, *Proc. Very Large Data Bases (VLDB)*, pp. 131-142, 2002.
- [9] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks, *Advance in cryptology, Eurocrypt'00*, pp. 139-155, 2000.
- [10] M. Bellare, P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, *ACM. Computer and Communication Security*, pp. 62-73, 1993.
- [11] Committee on Maintaining Privacy and Security in Health Care Applications of the National Information Infrastructure. For the Record: Protecting Electronic Health Information, National Academy Press, Washington, D.C., 1997.
- [12] V. S. Dimitrov, G. A. Jullien, and W. C. Miller. Complexity and fast algorithms for multi-exponentiations, *IEEE Transactions on Computers*, vol 49, no 2, pp. 141C147, 2000.
- [13] R. Dhamija, J. D. Tygar. Phish and HIPs: Human Interactive Proofs to Detect Phishing Attacks, *International Workshop on Human Interactive Proofs*, LNCS 3517, pp. 127-141, 2005.
- [14] W. Ford, B. S. Kaliski Jr. Sever-assisted Generation of a Strong Secret From a Password, *IEEE. 9th International Workshop on Enabling Technologies*, 2000.
- [15] L. Gong, M. Lomas, R. Needham, J. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks, *IEEE Journal on Selected Areas in Communications*, 11(5), pp. 648-656, 1993.
- [16] O. Goldreich. Secure Multi-party Computation, Working Draft, Version 1.3, June 2001.
- [17] R. Ganesan. Yaksha: Augmenting Kerberos with Public Key Cryptography, *Symposium on Network and Distributed System Security (SNDSS'95)*, pp. 132-143, 1995.
- [18] S. Halevi, H. Krawczyk. Public-key Cryptography and Password Protocols, *ACM. Transactions on Information and System Security Computer (TISSEC)*, 2(3), pp. 230-268, 1999
- [19] D. P. Jablon. Password Authentication Using Multiple Servers, *RSA Security Conference*, LNCS 2020, pp. 344-360, 2001.

- [20] D. V. Klein. Foiling the Cracker - A Survey of, and Improvements to, Password Security, *2nd USENIX Security*, pp. 5-14, 1990.
- [21] 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.
- [22] J. Katz, R. Ostrovsky, M. Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords, *Advances in Cryptology, Eurocrypt'01*, LNCS 2045, pp. 475-494, 2001.
- [23] J. Katz, R. Ostrovsky, M. Yung. Forward Secrecy in Password-Only Key Exchange Protocols, *Proc. Security in Communication Networks*, 2002
- [24] P. Mackenzie, T. Shrimpton, M. Jakobsson. Threshold Password-Authenticated Key Exchange, *Advances in Cryptology, Crypto'02*, LNCS 2442, pp. 385-400, 2002.
- [25] M. D. Raimondo, R. Gennaro. Provably Secure Threshold Password-Authenticated Key Exchange. *Advances in Cryptology, Eurocrypt'03*, LNCS 2656, pp. 507-523, 2003.
- [26] Y. J. Yang, F. Bao, and R. H. Deng. A New Architecture for Authentication and Key Exchange Using Password for Federated Enterprises, *20th IFIP International Information Security Conference, SEC'05*, 2005.