An Efficient Password-Only Two-Server Authenticated Key Exchange System*

Haimin Jin[1,2], Duncan S. Wong[1], and Yinlong Xu[2]

[1] Department of Computer Science
City University of Hong Kong
Hong Kong, China
duncan@cityu.edu.hk
[2] Department of Computer Science
University of Science and Technology of China
China
jhm1213@mail.ustc.edu.cn, ylxu@ustc.edu.cn

Abstract. One of the prominent advantages of password-only two-server authenticated key exchange is that the user password will remain secure against offline dictionary attacks even after one of the servers has been compromised. The first system of this type was proposed by Yang, Deng and Bao in 2006. The system is efficient with a total of eight communication rounds in one protocol run. However, the security assumptions are strong. It assumes that one particular server cannot be compromised by an active adversary. It also assumes that there exists a secure communication channel between the two servers. Recently, a new protocol has been proposed by the same group of researchers. The new one removes these assumptions, but in return pays a very high price on the communication overhead. It takes altogether ten rounds to complete one protocol run and requires more computation. Therefore, the question remains is whether it is possible to build a protocol which can significantly reduce the number of communication rounds without introducing additional security assumptions or computational complexity. In this paper, we give an affirmative answer by proposing a very efficient protocol with no additional assumption introduced. The protocol requires only six communication rounds without increasing the computational complexity.

1 Introduction

Password-only authenticated key exchange is a scheme which allows a user who holds only a low-entropy password to conduct authentication and key exchange with a server. Comparing with related types of authenticated key exchange schemes, for example, schemes based on cryptographic keys, password-only authenticated key exchange is very practical with high usability, because users only need to memorize a short password which is already used commonly in existing authentication systems.

* The work was supported by CityU grants (Project Nos. 7001844, 7001959, 7002001).

S. Qing, H. Imai, and G. Wang (Eds.): ICICS 2007, LNCS 4861, pp. 44–56, 2007.

A **Password-only Two-Server Authenticated Key Exchange** (PTAKE) scheme [18,17] is an extension of the conventional single-server setting [13]. Besides a user and a server, PTAKE also has an additional server. The existing, front-end, server that the user is communicating with is called the *Service Server*, SS, and the additional, back-end, server which communicates only with SS is called the *Control Server*, CS. In a conventional single-server scheme, the server has a database of the users' passwords or some verification data of the passwords. If the server is compromised, an adversary can obtain the passwords of all the users directly from the database, or be able to launch offline dictionary attacks against all users' passwords as the database provides enough information about the passwords. This problem is generally referred to as *single point of failure*.

In a PTAKE scheme instead, the password of each user is split into two shares and each server is given only one share, so that the password cannot be obtained in an information theoretic sense if only one share is known. Currently, all the concrete PTAKE schemes [18,17] use the same splitting mechanism: two random shares π_1 and π_2 are generated such that the password $\pi \equiv \pi_1 + \pi_2 \pmod{q}$ for a large prime q. A secure PTAKE scheme is designed such that even if one of the two servers is compromised, the scheme should still be able to thwart offline dictionary attacks against π. This feature together with the architecture of PTAKE yields a very desirable and practical system. As we can see, users only interact with SS but will never interact directly with CS, while CS only interacts with SS. This creates two networks, one *external* and one *internal*, with SS acting as the bridge/firewall between these two. In practice, this makes an outsider very difficult to compromise the internal network. The internal network can also be totally hidden from the outsiders. It is even possible for us to make outsiders totally unaware of the existence of CS. In addition, since CS only interacts with SS, it is relatively easy to provide maximal security protection for CS and make it much more difficult to compromise. To defend against insider attacks, it is also much easier to do so than the conventional one-server scheme as the administrative and operational tasks of SS and CS are separated and can be carried out by two independent teams. They do not share any secret.

The first PTAKE was proposed by Yang, Deng and Bao [18] in 2006. In one protocol run of their scheme, the user carries out four rounds of communications with SS and the CS carries out another four rounds with SS. The total number of communications rounds is therefore eight. Following the notion of external and internal networks, they assume that the internal network is impossible for an active adversary[1] to compromise. This assumption makes certain sense for some systems, but not in general. An insider who has full access to CS but not to SS may collude with an outsider and launch a successful offline dictionary attack against their scheme. Another issue of the scheme is that the communication channel between the two servers needs to be secure against eavesdroppers in order to ensure the security of the session key. This also introduces some additional cost for actual implementation of the scheme.

[1] An active adversary is considered to be an adversary which can do both eavesdropping and hijacking of messages.

Recently, a new scheme has been proposed by the same group of researchers [17]. The new one solves both of the problems mentioned above. In addition, it further enhances the security with respect to the session key, namely, CS can no longer compute the session key established between the user and SS. This is a desirable feature in practice as SS (Service Server) is generally the one to provide actual services while CS is simply called in for authenticating the user. There is no need or it simply downgrades the user's privacy if we let CS know the session key. The tradeoff of this new scheme is that the number of communication rounds for completing one protocol run is increased to ten.

1.1 Our Results

For the latest conventional password-only single-server schemes [13], the number of communication rounds between the user and the server is usually only three. In the two-server setting, there are three parties. The question we are asking is whether we can build a secure PTAKE scheme which takes only three communication rounds for mutual authentication between the user and SS, another three rounds for mutual authentication between CS and SS, and piggybacks some additional messages along these rounds for mutual authentication between the user and CS. As a result, the scheme only requires six rounds of communications to complete one protocol run. It is also desirable if we can attain the same level of security as in [17], namely

1. the scheme remains secure against offline dictionary attacks after one of CS and SS has been compromised by an *active* adversary;
2. no secure channel is required between CS and SS;
3. at the end of a protocol run, for any of the three parties (the user, SS and CS), the party can ensure that the other two parties have been involved; and
4. an *honest-but-curious* CS cannot compute the session key.

In this paper, we give an affirmative answer to this question. We propose a very efficient protocol which requires only six communication rounds with no additional assumption introduced, and satisfies all the security requirements above.

1.2 Related Work

Passwords chosen by users generally have very low entropy. Dictionary attacks are feasible by efficiently enumerating all the possible passwords from a dictionary, if enough information is given to an attacker. Dictionary attacks can be launched *online* or *offline*. In an online attack, an attacker attempts to logon to a server by trying all possible passwords until a correct one is found. Usually, this can easily be defended against at the system level by designing a system such that an unsuccessful online attack is *detectable* and limiting the number of unsuccessful login attempts. In an offline attack, the attacker records several successful login sessions and then tries all the possible passwords against the login transcripts. This type of attacks is notoriously difficult to defend against

and it is the main challenge for designing a secure password-based authenticated key exchange scheme.

On the design of password-based authenticated key exchange schemes, most of the schemes (e.g. most of the schemes in [13]) belong to the *single-server* category. As explained, single-server setting has a serious drawback which is called single point of failure. For enhancing the robustness, *two-server* and *multi-server* models have been proposed [12,14,15,16]. For multi-server schemes, although they alleviate the robustness problem, they usually require a user to communicate simultaneously with multiple servers or have the protocols become very expensive, especially when too many servers are getting involved.

The two-server setting seems to facilitate a very good balance between robustness and protocol complexity. Recently, a practical two-server architecture is proposed by Yang, Deng and Bao [18,17]. This architecture is a password-only variant of the one introduced by Brainard et al. [8]. As described in the Introduction section of this paper, the user communicates only with the Service Server (SS) while the Control Server (CS) communicates with SS only. In this paper, we propose a scheme which achieves the same security level as that of [17] (summarized in Sec. 1.1) but with much fewer number of communication rounds. Our scheme has even fewer rounds than the weaker scheme proposed in [18].

Organization. In the next section, we describe the basic tools that are used in our PTAKE scheme and then give the full details of the scheme. In Sec. 3, we analyze its security in terms of off-line dictionary attacks, pairwise mutual authentication and key privacy against Control Server. In Sec. 4, we provide the performance analysis and compare our protocol with Yang, Deng and Bao's in [17]. In Sec. 5, we conclude and mention some of our future work.

2 A New Efficient PTAKE

As introduced in Sec. 1, a protocol run of PTAKE involves a user U, a Service Server SS and a Control Server CS. There is a communication link between U and SS, another link between SS and CS, but no direct link between U and CS.

We consider two types of adversaries: passive adversaries and active adversaries. A passive adversary can eavesdrop any of the channels and try to derive user passwords from protocol transcripts. If a server is compromised by the passive adversary, all the internal states, which include the password share, of the server will be accessible by the adversary, but the server will still behave according to the protocol specification. An active adversary controls all the communication channels. If a server is compromised by an active adversary, all internal states of the server will be known to the adversary and the adversary has full control on how the server behaves.

Note that all literature [18,17] in this field assumes that the servers do not support multiple simultaneous sessions for one single user. We believe that this assumption is realistic as it usually indicates identity theft when multiple sessions from the same user are initiating from different places, for example, in the

e-banking applications. We leave the study of simultaneous multiple session setting as our future work.

2.1 Registration and Initialization

To begin with, user U first needs to register to the servers through some out-of-band channels. In this paper, we assume that U has already generated random password shares π_1 and π_2 such that U's password π is congruent to $\pi_1 + \pi_2 \pmod{q}$ for some large prime q. We refer readers to [18] for details on how this registration and initialization phase can be carried out.

In the following, we let g_1 and g_2 be distinct random generators of some multiplicative group G of large prime order q. Assume that the discrete logarithm problem in G is intractable, and also the discrete logarithm of g_2 to g_1 is intractable.

2.2 Basic Techniques

Before describing our proposed PTAKE, in this section, we propose several primitive techniques that we use repeatedly in our protocol. These techniques are mainly for defending against offline dictionary attacks while carrying out key exchange.

Since the password $\pi \equiv \pi_1 + \pi_2 \pmod{q}$, our protocol has to prevent an active adversary from obtaining any information related to $\pi_1 + \pi_2$ even under the condition that the adversary has already known one of these two password shares. There are two basic building blocks in our design.

The first building block is applying a *blinding factor* to the password or a password share. This building block has the message form of $M_1 = g_1^r g_2^x$. Here x is the password or a password share (i.e. π, π_1 or π_2), and r is chosen uniformly at random from $\mathbb{Z}_q^*$. From M_1, the adversary cannot get anything useful for launching offline dictionary attacks for getting x (from g_2^x) as g_1^r is not known. Table 1 shows some examples of this building block being used in our protocol which will be described shortly.

Table 1. Examples of the First Building Block

U $\rightarrow$ SS: $B = g_1^a g_2^\pi$	g_1^a is the blinding factor
SS $\rightarrow$ CS: $B_1 = B/(g_1^{b_1} g_2^{\pi_1})$	$g_1^{b_1}$ is the blinding factor
CS $\rightarrow$ SS: $B_4 = g_1^{b_4} g_2^{\pi_2}$	$g_1^{b_4}$ is the blinding factor

The second building block is the *randomization* of the messages received which are in the form of $M_2 = (D/g_2^x)^r$, $M_3 = g_1^r$. Here $r \in_R \mathbb{Z}_q^*$, x is a password share (but not the password), D is the message received.

Note that D can be generated by the adversary. This implies that the adversary is able to compute $(g_2^x)^r$ from M_2 and M_3. To see this, suppose the adversary sets $D = g_1^d$ for some arbitrarily picked d. The adversary can compute

$(g_2^x)^r = M_3^d/M_2$. Since r is chosen uniformly at random from $\mathbb{Z}_q^*$ and is unknown to the adversary, knowing g_1^r and g_2^{rx} does not help the adversary compute x as the adversary cannot determine the value of g_2^r under the assumption that DDH problem is hard[2]. Table 2 lists some examples on how this building bock is used in our protocol.

Table 2. Examples of the Second Building Block

$\mathsf{CS} \rightarrow \mathsf{SS}$: B_2, A_1 $B_2 = (B_1/g_2^{\pi_2})^{b_2}$, $A_1 = g_1^{b_2}$
$\mathsf{SS} \rightarrow \mathsf{U}$: B_5, A_3 $B_5 = ((Bg_1^{b_6}/B_4)/g_2^{\pi_1})^{b_5}$, $A_3 = g_1^{b_5}$

2.3 The PTAKE Protocol

The protocol is described in Fig. 1.

To initiate a request for service, U selects a random $a \in_R \mathbb{Z}_q^*$ and computes $B = g_1^a g_2^{\pi}$, then sends request Req, identity U and B to SS in $M1$. Upon receiving $M1$, SS selects a random $b_1 \in_R \mathbb{Z}_q^*$, computes $B_1 = B/(g_1^{b_1} g_2^{\pi_1}) = g_1^{a-b_1} g_2^{\pi_2}$, then sends U, SS and B_1 to CS in $M2$. Upon receiving $M2$, CS selects $b_2, b_4 \in_R \mathbb{Z}_q^*$, computes $B_2 = (B_1/g_2^{\pi_2})^{b_2} = (g_1^{a-b_1})^{b_2}$, $A_1 = g_1^{b_2}$ and $B_4 = g_1^{b_4} g_2^{\pi_2}$, then sends A_1, B_2 and B_4 back to SS in $M3$. SS selects $b_3, b_5, b_6 \in_R \mathbb{Z}_q^*$ and computes $B_3 = (B_2 A^{b_1})^{b_3} = g_1^{ab_2b_3}$, $S_1 = h(B_3)$, $A_2 = A_1^{b_3}$, $B_5 = (B/(B_4 g_2^{\pi_1}))^{b_5} g_1^{b_6 b_5} = g_1^{(a-b_4+b_6)b_5}$ and $A_3 = g_1^{b_5}$, then sends S_1, A_2, B_5, A_3 to U in $M4$. Upon receiving $M4$, U checks if $S_1 \stackrel{?}{=} h(A_2^a)$: if true, U accepts and computes $S_2 = h(A_2^a, 0)$. It then selects a random $a^* \in_R \mathbb{Z}_q^*$, computes $A_4 = A_3^{a^*}$, $B_6 = (A_3^a/B_5)^{a^*}$ and sends S_2, B_6, A_4 to SS in $M5$. The session key $K_u = h(A_2^a, \mathsf{U}, \mathsf{SS})$ is also computed. Otherwise, U aborts. Upon receiving $M5$, SS checks whether $S_2 \stackrel{?}{=} h(B_3, 0)$: if true, SS accepts, computes $S_3 = h(B_6 A_4^{b_6})$ and session key $K_{ss} = h(B_3, \mathsf{U}, SS)$, and sends S_3, A_4 to CS in $M6$. Otherwise, SS aborts. Upon receiving $M6$, CS checks whether $S_3 \stackrel{?}{=} h(A_4^{b_4})$, if it does not hold, CS aborts. Otherwise, CS accepts. In all the steps above, when an element of G is received, the party should always check if the element is not equal to 1. If so, the protocol should be aborted.

3 Security Analysis

Recall that one of the main goals of our proposed scheme is to resist off-line dictionary attacks by an active adversary who has compromised one of the two servers. In the following, we first examine the proposed scheme against a compromised $\mathcal{CS}$ and then a compromised $\mathcal{SS}$. We do not need to consider the case

[2] DDH assumption: given (g, g^r, h, z) where $g, h \in_R G$, $r \in_R \mathbb{Z}_q$, determine if $z = h^r$ or just an element chosen uniformly at random from G.

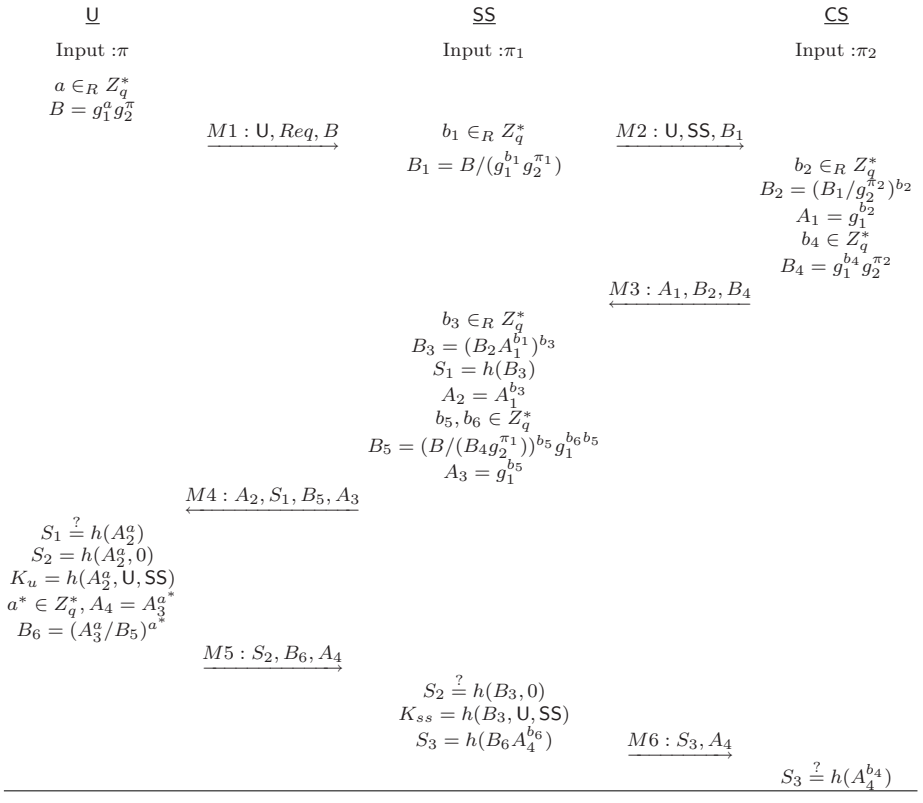

Fig. 1. Our Proposed Efficient PTAKE

that an active adversary does not have any server compromised, as this scenario has already been considered in the first two scenarios. After evaluating the security against off-line dictionary attacks, we will show that the scheme satisfies the authentication requirement as well as the desirable session key privacy, especially against the *Honest-but-Curious* CS.

3.1 Security Against Off-Line Dictionary Attacks

So far for PTAKE, there is no formal security model defined with comparable formality to [2,9] for conventional single-server password-based authenticated key exchange or [4,5,1,10] for cryptographic key based authenticated key exchange schemes. We believe that it is important to propose a formal security model for PTAKE and we consider this to be our next work. In this paper, our focus is on the performance optimization while providing the proposed scheme with some heuristic security evidence which is comparable to that in [17].

Also note that we follow the current literature [18,17] in this field by assuming that the servers do not support multiple simultaneous sessions from one single user. If a user is attempting to make simultaneous logins, the servers should consider this as some impersonation attacks. For conventional single-server password-based authenticated key exchange, multiple sessions are generally considered in the corresponding models [2,9]. For preventing various kinds of interleaving attacks [6,7], "binding" mechanisms of the identities of senders and receivers as well as the session IDs are generally required. Under the current single-session setting, the mechanisms may not be needed. Also because of this weaker security requirement, our scheme can achieve such a great efficiency. We believe that it is interesting to construct a provably secure PTAKE without sacrificing its efficiency when compared with our proposed scheme in this paper.

Proposition 1. *The proposed scheme can defend against offline dictionary attacks launched by an active adversary which has compromised* CS.

Through eavesdropping and hijacking the communication channels, we can see that the honest parties, U and SS, are actually providing the following oracles to the active adversary:

- Oracle1: It outputs $B = g_1^a g_2^\pi$, where a is chosen uniformly at random from $\mathbb{Z}_q^*$ and π is the password of U.
- Oracle2: On input $B, X, Y, Z \in G$, it outputs a quadruple: $B/(g_1^{b_1} g_2^{\pi_1})$, X^{b_3}, $h((YX^{b_1})^{b_3})$, $(B/(Zg_2^{\pi_1}))^{b_5} g_1^{b_5 b_6}$, where $b_1, b_3, b_5, b_6 \in_R \mathbb{Z}_q^*$ and π_1 is SS's password share.
- Oracle3: On input $U, V, W, B \in G$ and another finite binary string S_1, it outputs $O_1 = h(W^a, 0)$, $O_2 = U^{a^*}$, $O_3 = (U^a/V)^{a^*}$ only if $S_1 = h(W^a)$, where a is the random number chosen by Oracle1 when outputting B. If Oracle1 has never outputted B before, Oracle3 returns $\perp$. Note that a^* above is chosen uniformly at random by Oracle3.
- Oracle4: On input $S, T, S_2, U, V, O_1, O_2, O_3$, it outputs $h(ST^{b_6})$ only if $h((VU^{b_1})^{b_3}, 0) = S_2$, where U, V, b_1, b_3 and b_6 are corresponding to one query of Oracle3. Note that O_1, O_2, O_3 are needed to be in the query for identifying which query of Oracle3 it is associated with.

Oracle1 outputs B in the form of first building block described in Sec. 2.2. As explained, it gives no additional information about π. Oracle2's output has forms of both first and second building blocks described in Sec. 2.2. Again, as explained, they give no additional information about π or π_1 if DDH problem is hard. For Oracle3 and Oracle4, no particular password related information is involved and therefore do not provide additional information about π or π_1.

Proposition 2. *The proposed scheme can defend against offline dictionary attacks launched by an active adversary which has compromised* SS.

Similarly, we may consider the honest parties, U and CS, to provide the following oracles to the active adversary:

- Oracle1′: It outputs $B = g_1^a g_2^\pi$, where a is chosen uniformly at random from $\mathbb{Z}_q^*$ and π is the password of U.

- Oracle2′: On input $X \in G$, it outputs a triple: $g_1^{b_2}$, $(X/g_2^{\pi_2})^{b_2}$, $g_1^{b_4} g_2^{\pi_2}$, where $b_2, b_4 \in_R \mathbb{Z}_q^*$ and π_2 is CS's password share.
- Oracle3′: On input $U, V, W, B \in G$ and S_1, it outputs $h(W^a, 0), U^{a^*}, (U^a/V)^{a^*}$ only if $S_1 = h(W^a)$, where a is the random number chosen by Oracle1′ when outputting B. If Oracle1′ has never outputted B before, Oracle3′ returns $\perp$. Note that a^* above is chosen uniformly at random by Oracle3′.

Oracle1′ outputs B in the form of first building block described in Sec. 2.2. It gives no additional information about π. The output of Oracle2′ has forms of both first and second building blocks described in Sec. 2.2. As explained, they give no additional information about π or π_2 if DDH problem is hard. For Oracle3′, no particular password related information is involved and therefore do not provide any additional information about π or π_2.

3.2 Authentication

Unlike one-way or mutual authentication in a two-party setting, a secure TPAKE should ensure that *for each* of the three parties (i.e. U, SS, CS), the party is given enough evidence of the involvement of the other two parties before the party can complete a protocol run without early abortion.

U *authenticates* CS *and* SS: The authentication idea is to have U send a 'masked' commitment of π, in the form of $B = g_1^a g_2^\pi$, to servers and require the servers to work jointly to remove the commitment g_2^π of π from B. If the returned value is g_1^a, the two servers are authenticated. However, it is not trivial to do so because all the communication channels are open and susceptible to both passive and active attacks. For making the authentication idea work, blinding factors and randomization techniques are introduced. We can see that SS computes $B_1 = g_1^{-b_1} B g_2^{-\pi_1}$ and CS computes $B_2 = (B_1 g_2^{-\pi_2})^{b_2}$, where the component $g_1^{-b_1}$ in the computation of B_1 is the blinding factor; and the power b_2 for computing B_2 is the randomization. The authentication idea remains the same, that is, $Bg_2^{-\pi_1}$ when computing B_1 and $B_1 g_2^{-\pi_2}$ when computing B_2, in other words, having SS and CS remove g_2^π from B using their knowledge of π_1 and π_2. Note that after adding the blinding factor and randomization, the value received by U becomes $h(B_3) = h(g_1^{a b_2 b_3})$ where b_3 is the randomization introduced by SS. This is to prevent a malicious CS from launching off-line dictionary attack against an honest SS.

Note that randomization is an important technique to defend against off-line dictionary attack while allowing an initiator (in this case, it is U) to authenticate. This can be seen by imagine that the two servers were one single party holding π. Upon receiving $B = g_1^a g_2^\pi$, the combined server computes $S_1 = h((B/g_2^\pi)^{b'})$ for a random $b' \in \mathbb{Z}_q^*$, and sends S_1 and $A_2 = g_1^{b'}$ back to U. U then checks if $S_1 \stackrel{?}{=} h(A_2^a)$. Suppose an adversary impersonates U and sets B as $g_1^{a'}$. The received response from the combined server will become $h(g_1^{a'b'} g_2^{-\pi b'})$. We can see that the adversary is not able to determine $g_2^{b'}$ from all the known values and therefore, is not able to launch off-line dictionary attack.

CS *authenticates* SS *and* U: The approach is similar to the above. The 'masked' commitment of π_2 is $B_4 = g_1^{b_4} g_2^{\pi_2}$. SS and U then work jointly to remove the commitment $g_2^{\pi_2}$ from B_4 using their knowledge of π_1 and π, respectively. After introducing blinding factors and randomization techniques, the value received by CS becomes $S_3 = h(g_1^{b_4 a^* b_5})$. With $A_4 = g_1^{a^* b_5}$, CS can check if $S_3 \stackrel{?}{=} h(A_4^{b_4})$.

As mentioned, the role of $\mathcal{CS}$ is to assist $\mathcal{SS}$ in authenticating $\mathcal{U}$. Therefore, it some security model, it may be fine if we remove the components corresponding to authenticating $\mathcal{SS}$ and $\mathcal{U}$ by $\mathcal{CS}$, that is, those related to B_4. However, in a more general security model, $\mathcal{SS}$ may make use of $\mathcal{CS}$ as an oracle for launching an *unlimited* and *undetectable* online dictionary attack for compromising $\mathcal{CS}$'s password share π_2. Specified in our protocol, $[B_4, B_5, B_6, A_3, A_4, S_3]$ are added to provide this authentication. If without checking $S_3 \stackrel{?}{=} h(A_4^{b_4})$ by $\mathcal{CS}$, $\mathcal{SS}$ can arbitrarily choose a trial password π_{guess} and send $B_1 = g_2^{\pi_{guess} - \pi_1}$ to $\mathcal{CS}$, then check if the received value B_2 from $\mathcal{CS}$, which should be of the form $(g_2^{\pi_{guess} - \pi_1} / g_2^{\pi_2})^{b_2}$ is equal to 1. Without S_3, we can see that $\mathcal{CS}$ has no way to find out if $\mathcal{SS}$ is launching this kind of online dictionary attacks. Therefore, $\mathcal{SS}$ can simply repeat the trial above until the value of π_2 is found. Without B_4 and the associated components for authenticating $\mathcal{SS}$ and $\mathcal{U}$, this type of online dictionary attacks is *undetectable* and therefore cannot be defended against using the conventional system-level method which is commonly applied to limit the number of unsuccessful login attempts (Sec. 1.2).

SS *authenticates* CS *and* U: The authentication idea is similar but with a different order. SS obtains a 'masked' commitment of π from U first. SS asks CS to work together for removing the commitment of π. If the 'masked' commitment received from the claimed U is properly formatted and CS also performs according to the protocol, the commitment of π will be removed, with the blinding factors and randomization remained as $S_2 = h(g_1^{a b_2 b_3}, 0)$. SS knows the value of b_3. From $B_2 = g_1^{(a - b_1) b_2}$, $A_1 = g_1^{b_2}$ and b_1, SS can compute $g_1^{a b_2}$. Hence SS can verify if $S_2 \stackrel{?}{=} h(g_1^{a b_2 b_3})$.

3.3 Key Privacy Against an Honest-But-Curious CS

In the following, we focus on discussing the key privacy against an honest-but-curious CS rather than an eavesdropper. This is because the key privacy against the honest-but-curious CS implies the key privacy against an eavesdropper.

Our idea is based on the Diffie-Hellman key exchange between U and SS. At the end of the protocol, we target to have each of U and SS generate a session key which is essentially computed from the Diffie-Hellman contributions of these two parties, while ensuring the CS is not able to get the discrete logarithm of any of the two contributions.

The session key established between U and SS is $h(g_1^{a b_2 b_3}, \mathsf{U}, \mathsf{SS})$, where b_2 is picked by CS. Due to the idealness assumption of h as a random oracle [3], CS has to know $g_1^{a b_2 b_3}$ in order to obtain the session key. This implies that CS has to know $g_1^{a b_3}$. However, the only available information related to $g_1^{a b_3}$ is g_1^a and

g^{b_3}. Hence CS has to solve the Computational Diffie-Hellman problem in order to obtain the session key.

4 Performance

We now compare the performance of our proposed scheme with the YDB scheme proposed by Yang, Deng and Bao [17] as their scheme is the only one currently known to also satisfy all the requirements stated in Sec. 1.1.

Let $|p|$ and $|h|$ denote the bit length of p and the output of hash function $h(.)$, respectively. The performance comparison is given in Table 3.

Table 3. Performance Comparison

		U	SS	CS
Computation (exponentiations)	YDB scheme	5/2	6/1	3/1
	our scheme	4/1	6/2	4/2
Communication (bits)	YDB scheme	$4\|p\| + 2\|h\|$	$8\|p\| + 3\|h\|$	$4\|p\| + \|h\|$
	our scheme	$6\|p\| + 2\|h\|$	$11\|p\| + 2\|h\|$	$5\|p\| + 1\|h\|$
Communication (rounds)	YDB scheme	6	10	4
	our scheme	3	6	3

Computational Complexity: Since the complexity of exponentiation dominates a party's computational overhead, we count the number of exponentiations required for each party. The digits before "/" in the table denote the total number of exponentiations performed by the party, the digits followed denote the number of exponentiations that can be pre-computed. Note that by leveraging on the techniques in [11], each of $g_1^a g_2^{\pi}$, $g_1^{b_1} g_2^{\pi_1}$, $(B_2 A^{b_1})^{b_3}$ and $(B_1/g_2^{\pi_2})^{b_2}$ can be computed by a single exponentiation operation. We can see that the computational complexity of our proposed scheme is comparable to that of YDB scheme.

Communication Performance in terms of Effective Bits: As $|Q|$ is only one bit longer than $|p|$, we do not distinguish them when evaluating the YDB scheme. The number of bits transmitted by each party is comparable to the YDB scheme. Note that this measures the total number of *effective* bits transmitted that are related to the protocol. It does not include the additional overhead of headers and trailers required for transferring the data packets in each communication round. We will see just in the next point that our proposed scheme has much fewer number of rounds than that of YDB scheme. Therefore, our scheme incurs much less communication overhead due to headers and trailers in actual implementation.

Communication Performance in terms of Rounds: One round is a one-way transmission of messages. Our proposed scheme has a total of six rounds while YDB scheme requires altogether 10 rounds to complete one protocol run. Also note that the number of rounds made by U is significantly reduced to half from the original number. This is desirable especially for low-power wireless users.

5 Conclusion and Future Work

We proposed a new PTAKE which outperforms all previously proposed ones in terms of the number of communications rounds, while maintaining almost the same extent of computational complexity.

This proposed scheme is particularly suitable for implementation on resource-constrained wireless devices. Transmitting radio signals on these devices usually consumes much more power than computation does. Furthermore, if we use appropriate elliptic curve groups in actual implementation, the computational requirement of our scheme can further be reduced. Therefore, our scheme which reduces the number of communication rounds by 40% helps reducing battery power consumption as well as improving the performance of actual implementation significantly.

While we examined the security of the proposed protocol, a formal treatment of the system is necessary. Currently, there is no formal security model proposed for PTAKE. Therefore, our future work is to formally define and validate the security of PTAKE and provide formal proofs with various desirable security features captured. Examples of desirable features are security against known-key attacks[3], forward secrecy[4], etc.

Acknowledgements

We would like to thank Yanjiang Yang for his valuable comments.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: A modular approach to the design and analysis of authentication and key exchange protocols. In: Proc. 30th ACM Symp. on Theory of Computing, pp. 419–428. ACM, New York (1998)
2. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, Springer, Heidelberg (2000)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: First ACM Conference on Computer and Communications Security, Fairfax, pp. 62–73. ACM, New York (1993)
4. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
5. Bellare, M., Rogaway, P.: Provably secure session key distribution – the three party case. In: Proc. 27th ACM Symp. on Theory of Computing, Las Vegas, pp. 57–66. ACM, New York (1995)

[3] An adversary should not be able to compromise the long-term secret (e.g. password) after obtaining some session keys.

[4] Previously established session keys should remain secure after all long-term secrets have been compromised.

6. Bird, R., Gopal, I., Herzberg, A., Janson, P., Kutten, S., Molva, R., Yung, M.: Systematic design of two-party authentication protocols. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 44–61. Springer, Heidelberg (1992)
7. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer, Heidelberg (2003)
8. Brainard, J., Juels, A., Kaliski, B., Szydlo, M.: A new two-server approach for authentication with short secrets. In: USENIX Security Symposium, pp. 201–214 (2003)
9. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally composable password-based key exchange. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005), `http://eprint.iacr.org/2005/196`
10. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001), `http://eprint.iacr.org/2001/040/`
11. Dimitrov, V.S., Jullien, G., Miller, W.C.: Complexity and fast algorithms for multiexponentiations. IEEE Transactions on Computers 49(2), 141–147 (2000)
12. Ford, W., Kaliski, B.: Server-assisted generation of a strong secret from a password. In: WETICE 2000: IEEE 9th International Workshop on Enabling, pp. 176–180. IEEE, Los Alamitos (2000)
13. IEEE. P1363.2 / D26: Standard Specifications for Password-based Public Key Cryptographic Techniques, (September 2006)
14. Jablon, D.P.: Password authentication using multiple servers. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 344–360. Springer, Heidelberg (2001)
15. MacKenzie, P., Shrimpton, T., Jakobsson, M.: Threshold password-authenticated key exchange. Journal of Cryptology 19(1), 22–66 (2006)
16. Raimondo, M.D., Gennaro, R.: Provably secure threshold password-authenticated key exchange. In: Biham, E. (ed.) EUROCRPYT 2003. LNCS, vol. 2656, pp. 507–523. Springer, Heidelberg (2003)
17. Yang, Y., Deng, R.H., Bao, F.: Fortifying password authentication in integrated healthcare delivery systems. In: ASIACCS 2006: Proc. the 2006 ACM Symposium on Information, Computer and Communications Security, pp. 255–265. ACM Press, New York (2006)
18. Yang, Y., Deng, R.H., Bao, F.: A practical password-based two-server authentication and key exchange system. IEEE Trans. Dependable and Secure Computing 3(2), 105–114 (2006)