SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

July 2017, Vol. 60 072103:1–072103:14
doi: 10.1007/s11432-016-5535-7

An efficient and practical threshold gateway-oriented password-authenticated key exchange protocol in the standard model

Fushan WEI[1,2]*, Jianfeng MA[1], Ruijie ZHANG[2], Chuangui MA[2] & Xuan WANG[3]

[1]*State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an* 710071*, China;*
[2]*State Key Laboratory of Mathematical Engineering and Advanced Computing,*
The PLA Information Engineering University, Zhengzhou 450001*, China;*
[3]*Engineering University of CAPF, Xi'an* 710078*, China*

Received October 30, 2015; accepted December 26, 2015; published online September 28, 2016

Abstract With the assistance of an authentication server, a gateway-oriented password-authenticated key exchange (GPAKE) protocol can establish a common session key shared between a client and a gateway. Unfortunately, a GPAKE protocol becomes totally insecure if an adversary can compromise the authentication server and steal the passwords of the clients. In order to provide resilience against adversaries who can hack into the authentication server, we propose a threshold GPAKE protocol and then present its security proof in the standard model based on the hardness of the decisional Diffie-Hellman (DDH) problem. In our proposal, the password is shared among n authentication servers and is secure unless the adversary corrupts more than $t+1$ servers. Our protocol requires $n > 3t$ servers to work. Compared with existing threshold PAKE protocols, our protocol maintains both stronger security and greater efficiency.

Keywords password, key exchange, gateway, threshold, provable security

Citation Wei F S, Ma J F, Zhang R J, et al. An efficient and practical threshold gateway-oriented password-authenticated key exchange protocol in the standard model. Sci China Inf Sci, 2017, 60(7): 072103, doi: 10.1007/s11432-016-5535-7

1 Introduction

With the rapid development of cloud computing technologies, more and more users can outsource their personal data to a cloud server to enjoy its ubiquitous services [1–3]. To protect the user's sensitive data from unauthorized access, user authentication and key exchange protocols are very important for access control and data transmission in cloud computing applications [4–6]. The password is the most popular mechanism for user authentication [7]. However, traditional password-authenticated key exchange (PAKE) protocols only deal with the "Client-Server" architecture. In order to bring PAKE protocols closer to practice, the notion of gateway-oriented password-authenticated key exchange (GPAKE) was put forward by Abdalla et al. in 2005 [8]. A GPAKE protocol can establish a common session key for a client and a gateway with the assistance of an authentication server. GPAKE protocols focus on

* Corresponding author (email: weifs831020@163.com)

the password-based authentication problem in the "Client-Gateway-Server" architecture, which takes into account the presence of gateways when clients communicate with authentication servers. Since the "Client-Gateway-Server" architecture is very popular in wireless communications, GPAKE protocols are suitable for mobile communication environments.

There are three security requirements for GPAKE protocols: semantic security of session keys, password protection against malicious gateways and key privacy with respect to honest-but-curious authentication servers. These security requirements make designing secure GPAKE protocols a non-trivial task. In 2006, Abdalla et al.'s GPAKE protocol [8] was found by Byun et al. [9] to be vulnerable to undetectable on-line dictionary attacks (UODA). A malicious gateway can randomly choose a password from the password dictionary and verify the correctness of its guess repeatedly, without being detected by the server. Byun et al. also suggested a countermeasure by using a message authentication code scheme to provide data integrity. Nevertheless, Kyung [10] demonstrated that Byun's solution was still vulnerable to the same attack. Kyung also presented an improved version called S-GPAKE by using a symmetric encryption scheme to mask the password information. In 2008, Abdalla et al. [11] put forward a stronger security model for GPAKE protocols and proposed an anonymous GPAKE protocol using private information retrieval protocols [12]. In 2010, Yoon et al. [13] identified several shortcomings of the S-GPAKE protocol and proposed their optimized GPAKE protocol. In 2012, Wei et al. [14] proposed the first GPAKE that was provably secure in the standard model by extending Jiang and Gong's protocol [15] to the gateway-oriented environment. In 2013, Wei et al. [16] pointed out serveral weaknesses of Yoon et al.'s protocol and then presented an improved protocol. Recently, Chien et al. [17] designed a provably secure GPAKE protocol that can resist password guessing attacks. Unfortunately, Choi et al. [18] soon found that Chien et al.'s protocol was actually insecure against off-line dictionary attacks.

Compared with PAKE protocols, GPAKE protocols provide a better protection to the authentication server since the gateway serves as a firewall to protect the authentication server from intrusion. However, GPAKE protocols become complitely insecure if the adversary manages to break the firewall, compromises the authentication server and finally steals all client passwords. GPAKE protocols, like PAKE protocols, are all vulnerable to this type of hacking attacks.

1.1 Threshold password-based authenticated key exchange

In centralized PAKE protocols, the authentication server stores all the passwords of the clients that are subscribed to the server in its database. If an adversary can corrupt or hack into the authentication server, he will be able to retrieve all of the password information of all the clients that are subscribed to the server. Even if the authentication server stores the verification value of the passwords, the adversary can still obtain the passwords by performing an off-line dictionary attack on the password verification data. Considering the scale and the frequency of the server compromises that occur through the Internet, hacking technologies have become a critical security challenge for PAKE protocols.

In order to provide some resilience against hacker attacks, Ford et al. [19] used the idea of sharing the password among several servers. They proposed a distributed n-out-of-n PAKE protocol which assumes the servers have public keys that are known to the clients. Since their protocol requires all of the servers to participate, it is not robust. Moreover, Ford et al. only presented heuristic arguments for their protocol. Jablon [20] quickly improved on [19]. His solution does not require the server's public key to be known to the clients, but again does not have any proof of security. MacKenzie et al. [21, 22] proposed the first provably secure (t, n) threshold PAKE protocol in the random oracle model. However, MacKenzie et al.'s solution is quite complicated because each server has its own local and global key pairs in its system. Subsequent to MacKenzie et al.'s work, Raimondo et al. [23, 24] also presented a threshold PAKE protocol in the standard model using the well-known Katz, Ovstrosky, and Yung protocol [25]. Their protocol cancels the assumption of servers' public keys and thus is more realistic. However, in Raimondo et al.'s protocol, the servers need to perform the distributed Pederson's verifiable secret sharing (VSS) scheme several times which makes their protocol inefficient from a practical point of view. Lee et al. [26] presented a threshold PAKE protocol using bilinear pairings for roaming services.

Table 1 Summary of differences between the protocols

Protocols	Architecture	Protocol type	Application scenarios	Advantages	Disadvantages
[8–11,13, 14,16–18]	Client-Gateway -Server	Gateway-oriented PAKE	Centralized password authentication with gateways	Efficient, closer to practice	Prone to UODA and hacker attacks
[19–24, 26–29]	Client -Servers	Threshold PAKE	Distributed password authentication	Robust and resilient to hacker attacks	Complicated inefficient
[30–33]	Client -Servers	Threshold PASS	Private data storage and retrieval	Robust and resilient to hacker attacks	Complicated inefficient

Chai et al. [27] also designed a threshold PAKE protocol for ad hoc networks. Unfortunately, Li et al. [28] demonstrated that Chai et al.'s protocol is insecure against the passive attacks, in which the attacker could collect secrets by eavesdropping on the communication channels and further derives the legal user's secret token in order to damage the security of system. Guo et al. [29] presented a threshold authentication architecture for wireless mesh networks. Recently, the notion of threshold password-authenticated secret sharing (TPASS) [30] was put forward owing to the need to securely store encrypted data in the cloud. Several TPASS schemes have been proposed [31–33]. But this is a new research topic for threshold PAKE protocols and much more research is needed. In the gateway-oriented setting, Abdalla et al. presented the first threshold GPAKE protocol in [8]. Instead of sharing the password, all of the servers in Abdalla et al.'s GPAKE protocol share a secret key for a public key encryption scheme under which all passwords are encrypted. Abdalla et al.'s threshold GPAKE protocol uses a zero-knowledge proof system due to Chaum and Pedersen [34], thus it is only provably secure in the random oracle model. In Table 1, we summarize the differences between these PAKE protocols.

1.2 Our contribution

In this paper, we pursue the line of the work in [8] and propose an efficient threshold GPAKE protocol, which is provably secure in the standard model based on the intractability of the decisional Diffie-Hellman (DDH) problem. To achieve this goal, we first put forward a more secure model for threshold GPAKE protocols, and then extend Jiang and Gong's protocol [15] to the threshold setting. In our proposal, the client basically executes a threshold PAKE protocol with $2t+1$ servers. A common secret value is established between the client and the $2t+1$ servers. The Diffie-Hellman key exchange materials of the client and the gateway are authenticated using the secret value. Thus, an efficient threshold PAKE protocol can also be derived from our proposal. The derived threshold PAKE is to Jiang and Gong's protocol [15] as the work of Raimondo and Gennaro [24] is to that of the Katz, Ovstrosky, and Yung protocol [25]. The most technically challenging part of our work is deciding how the servers can prove the knowledge of the password without ever reconstructing it. To prove the knowledge of the password in Jiang and Gong's protocol, the server encrypts the password using a chosen ciphertext attack (CCA) secure public key encryption scheme. However, this technique does not work in the distributed setting because none of the servers knows the password in threshold PAKE protocols. In order to solve this problem, we adopt the method of Katz et al. [35] when designing secure a universally composable-secure one round PAKE protocol. We let each server include as part of its message an encryption of its local hash key along with a proof that the ciphertext encrypts the local hash keys, which is consistent with its projective hash key. We can then reconstruct the hash key to verify the validity of the password.

Our protocol has several advantages. First, the security of our protocol does not assume idealizing assumptions and is conducted in the standard model under the simple and standard DDH assumption. Second, Our protocol provides stronger security by ensuring that the session key and the password are still secure even though the adversary corrupts up to t servers. Besides, mutual authentication between the client side and the server side is achieved, so our protocol can be proven secure against undetectable on-line dictionary attacks, which are a critical threat to GPAKE protocols. Last but not least, we keep the zero-knowledge proof to a minimum and hence our protocol is very efficient in terms of on-line computation. The servers only need to execute four distributed Pederson VSS schemes, which makes our protocol very practical.

The organization of the paper is as follows. In Section 2, we present the security model for threshold GPAKE protocols. In Section 3, we introduce some building blocks. We describe our protocol and prove its security in Section 4. We show that a secure threshold PAKE protocol can be derived from our protocol in Section 5. In Section 6, we give the performance analysis of our protocols. We conclude this paper in Section 7.

2 Security model

In this section, we introduce the security model for threshold GPAKE protocols, based on [8, 23].

2.1 Security model

A threshold GPAKE can establish a common session key for a client and a gateway with the assistance of n authentication servers. The user's password is shared among the servers for authentication. The user authenticates himself to the servers using his password while n authentication servers collaborate to authenticate the client. If an execution of the protocol is successful, the user will share a session key with the gateway such that the session key is unknown to anybody (especially the authentication servers) except two of them.

Protocol participants. There are three types of protocol participants in threshold GPAKE protocols: the client $C \in \mathcal{C}$, the gateway $G \in \mathcal{G}$ and n authentication servers $\{S_1, \ldots, S_n\}$ with $S_i \in \mathcal{S}$. Each client $C \in \mathcal{C}$ holds a password pw_C. Each server $S_i \in \mathcal{S}$ holds a vector of passwords $pw_{S_i} = \langle pw_{C_i} \rangle_{C \in \mathcal{C}}$ with an entry for each client, where pw_{C_i} is a share of user C's password pw_C.

Communication. We assume that the client and the gateway communicate with each other through an insecure public channel that is under the control of the adversary. However, the channel connecting the gateway to the servers is authenticated and private. For simplicity, we assume the gateway only interacts with a special authentication server, called the combiner. The servers are connected by a complete network of private point-to-point channels. The servers also have access to a broadcast channel such that if server S_i broadcasts a message, it is received by every other server and recognized as coming from S_i. We also assume the communication channels among the servers provide a partially synchronous message delivery, which means messages sent during the protocol will be received by their recipients within some fixed time bound.

Adversarial model. In threshold PAKE protocols, two types of adversaries are considered. A Type I adversary acts as an outside attacker. In this case, the adversary cannot distinguish the real session key from a random number and should not get any information of the password from protocol executions. A Type II adversary can corrupt at most t servers. In such a case, the session keys established between the client and the corrupted servers are known to the adversary, but the clients' passwords are still unknown to the adversary, even if it controls up to t servers. In threshold GPAKE protocols, the session key is established between the client and the gateway, so even the Type II adversary should not known the session key. In the adversarial model, here we consider a rather powerful adversary. The adversary $\mathcal{A}$ is allowed to compromise up to t servers and has complete control of the network (except for the secure channels of uncorrupted servers). However, the choice of the corrupted servers is static, which means the set of corrupted servers should be decided in advance.

Each participant can run several number of protocol instances concurrently. Let U^i denote the instance i of a participant U. We model the attack abilities of the adversary $\mathcal{A}$ through the following oracle queries.

• Execute$(C^i, G^j, (S_{n1}^{l_{n1}}, \ldots, S_{nk}^{l_{nk}}))$. This query models $\mathcal{A}$'s passive attack ability. $\mathcal{A}$ eavesdrops a protocol execution among a client instance C^i, a gateway instance G^j and k server instances, where k is the number of servers with $2t+1 \leqslant k \leqslant n$. $\mathcal{A}$ finally gets all the exchanged messages during the execution of the protocol.

• Send(U^i, m). This query enables $\mathcal{A}$ to send a faked message to instance U^i in the name of another participant. This query models $\mathcal{A}$'s active attack ability. The instance U^i will send back to $\mathcal{A}$ the message it will generate upon receiving m.

• Reveal(C^i/G^j). This query is used to capture the known key attack. Through this query, $\mathcal{A}$ receives the session key of either the client instance C^i or the gateway instance G^j.

• Test(U^i). This query does not captures $\mathcal{A}$'s real attack ability. It measures the session key security of instance U^i. The simulator flips a coin, and if the random bit is $b = 1$, it sends the real session key to the $\mathcal{A}$. Otherwise, the simulator sends a random key of the same size to $\mathcal{A}$. $\mathcal{A}$ will guess the value of b to win the game.

• TestPair(C^i, G^j). This query does not captures $\mathcal{A}$'s real attack ability either. It measures the key privacy of the session key shared between the client instance C^i and the gateway instance G^j against honest-but-curious servers. The simulator flips a coin, and if the random bit is $b = 1$, it sends the real session key to $\mathcal{A}$. Otherwise, the simulator sends a random key of the same size to $\mathcal{A}$. $\mathcal{A}$ will guess the value of b to win the game.

2.2 Security notions

We recall some security definitions in this section. The definitions of session identifications, partner identifications and partnering are remain as usual. We omit these definitions for simplicity. For more details, refer to [8].

Definition 1 (Freshness). An instance of a client or a gateway is said to be fresh if: (1) it has accepted; (2) there has been no Reveal query to the instance or its partner.

The definition of freshness is used to restrict the adversary from winning the attack game trivially by sending Test queries to instances whose session keys are known to the adversary. A threshold GPAKE protocols should provide security for session keys as well as for passwords. The session key established for the client and the gateway should be unknown to anyone except the two of them. The password should be unknown to the adversary unless it corrupts more than t servers. To capture these security requirements, we will present the following security definitions.

Semantic security. Suppose $\mathcal{P}$ is a threshold GPAKE protocol and $\mathcal{A}$ is a probabilistic polynomial time (PPT) adversary against the semantic security of $\mathcal{P}$. The adversary $\mathcal{A}$ is allowed to corrupt and control at most t servers. Moreover, we give $\mathcal{A}$ oracle access to Execute, Send, Corrupt and Test oracles. However, $\mathcal{A}$ can only make a Test query on fresh instances. If the adversary can correctly guess the random bit b used by the Test oracle, then we say the adversary succeeds in this attack game. We denote this event by Succ.

Definition 2. The advantage of an adversary $\mathcal{A}$ in winning the attack game for semantic security against the protocol $\mathcal{P}$ for semantic security, when passwords are uniformly drawn from a dictionary $\mathcal{D}$, is defined as

$$\mathrm{Adv}^{\mathrm{ake}}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) = 2 \cdot \Pr[\mathrm{Succ}] - 1.$$

We say a threshold GPAKE protocol $\mathcal{P}$ is semantically secure if $\mathrm{Adv}^{\mathrm{ake}}_{\mathcal{P},\mathcal{D}}(\mathcal{A})$ is only negligibly larger than $kQ_{\mathrm{send}}/|\mathcal{D}|$ for all PPT adversaries, where Q_{send} is number of Send queries and k is a constant.

Key privacy. The notion of key privacy captures the security of the session key with respect to honest-but-curious authentication servers, which know all the passwords of the clients. However, the servers are only allowed to perform passive attacks because the servers can trivially discover the session key from the knowledge of the password. We give the servers access to Execute and TestPair oracles. If the servers can correctly guess the random bit b used by the TestPair oracle, then we say they succeed in this attack game. Note that the TestPair oracles are restricted to session keys generated by two oracles.

Suppose $\mathcal{P}$ is a threshold GPAKE protocol and $\mathcal{A}$ is a PPT adversary against key privacy of $\mathcal{P}$. The advantage of $\mathcal{A}$ in breaking key privacy of the threshold GPAKE protocol $\mathcal{P}$ ($\mathrm{Adv}^{\mathrm{ake-kp}}_{\mathcal{P},\mathcal{D}}(\mathcal{A})$) can be defined in a similar manner as in definition 2. We say a threshold GPAKE protocol $\mathcal{P}$ achieves key privacy if $\mathrm{Adv}^{\mathrm{ake-kp}}_{\mathcal{P},\mathcal{D}}(\mathcal{A})$ is negligible for all PPT adversaries.

Server password protection. We consider a malicious gateway $\mathcal{A}$ who tries to guess the correct password by interacting with the servers. If a failed guess is not detected by the servers, then we say the malicious gateway is successful. Let $\mathrm{Adv}^{\mathrm{ake-uoda}}_{\mathcal{P},\mathcal{D}}(\mathcal{A})$ denote the success probability of the gateway.

Definition 3. A threshold GPAKE protocol $\mathcal{P}$ can resist undetectable on-line dictionary attacks if $\mathrm{Adv}^{\mathrm{ake-uoda}}_{\mathcal{P},\mathcal{D}}(\mathcal{A})$ is negligibly larger than $kQ_{\mathrm{send}}/|\mathcal{D}|$, where Q_{send} is number of active sessions and k is a constant.

3 Building blocks

3.1 Pederson's verifiable secret sharing scheme

Our protocol uses the well-known VSS scheme by Pedersen [36] which provides theoretic secrecy for the shared secret. Assume p, q are large primes with $q|(p-1)$, and $\mathbb{G}$ is the (unique) multiplicative subgroup of F_p^* of order q. Let g, h be elements of $\mathbb{G}$ where $\log_g h$ is unknown. To share a secret s in Z_q, a dealer first chooses two t-degree random polynomials $f(x)$ and $d(x)$ from $Z_q[x]$ such that $f(0) = s$. Let r denote the random free term of $d(x)$. The dealer sends $s_i = f(i) \bmod q$ and $r_i = d(i) \bmod q$ to player i via a private channel. Finally, the dealer broadcasts $E_k = g^{a_k}h^{b_k}$ for $k = 0, \ldots, t$ where a_k and b_k are coefficients of the k-degree term of $f(x)$ and $d(x)$ respectively. Note that s and r are committed to E_0 as $E_0 = g^s h^r$. The player i can verify the correctness of his share (s_i, r_i) by checking the equation:

$$g^{s_i}h^{r_i} = \prod_{k=0}^{t} (E_k)^{i^k}. \tag{1}$$

The players who hold shares that do not satisfy the above equation will broadcast a complaint. If more than t players complain, the dealer is disqualified.

To construction the secret, at least $t + 1$ players are required to reveal both s_i and r_i. The validity of the shares can be verified by Eq. (1). The secret s can be reconstructed by Lagrange interpolation. We use the following notation to denote an execution of Pederson's VSS:

$$\mathrm{PedVSS}(s, r)[g, h] \xrightarrow{f,d} (s_i, r_i)(E_0, \ldots, E_t). \tag{2}$$

To avoid the use of the trusted dealer, the players can share a random number unknown to any set of players less than $t + 1$ by using Pederson's verifiable secret sharing. The idea is that each player plays the role of the trusted dealer and runs a copy of Pederson's VSS with a random secret. The resulting shared secret is the sum of the secrets shared by each player. Note that by adding up the shares that he received, each player will hold a share of the final secret. Similarly, the verification for the final secret can also be computed by multiplying the commitments published by each player. We use the following notation to denote this distributed Pederson's verifiable secret sharing:

$$\mathrm{RandPedVSS}([s], [r])[g, h] \xrightarrow{f,d} (s_i, r_i, T_S)(E_0, \ldots, E_t). \tag{3}$$

The notation follows the same syntax as the basic Pederson's VSS, except that we add the value T_S which is the transcript of the n Pederson's VSS schemes executed by each player. All the parameters refer to the final secret. $[s]$ and $[r]$ imply that the random numbers are unknown to any set of players less than $t + 1$.

3.2 Shared exponentiation of secrets

We also use Raimondo et al.'s protocol for exponentiation of shared secrets [23]. Let $s_1, \ldots, s_m$ be m secrets shared among n servers by distributed Pederson VSS schemes. Raimondo et al.'s protocol for exponentiation of shared secrets can be used to compute $g_1^{s_1} \ldots g_m^{s_m}$ with any combination of basis (i.e. the basis used to exponentiate the secrets need not be equal to the ones used in the distributed Pederson VSS schemes). The execution of the protocol will not reveal any information about the secrets $s_1, \ldots, s_m$ beyond the output value $g_1^{s_1} \ldots g_m^{s_m}$. We will refer to an execution of Raimondo et al.'s protocol with the following notation:

$$\text{Shared-Exp}[T_{s_1}, \ldots, T_{s_m}](g_1, \ldots, g_m) \rightarrow g_1^{s_1} \ldots g_m^{s_m}, \tag{4}$$

where T_{s_i} is the transcript of Pederson's VSS scheme when sharing the secret s_i.

4 Threshold GPAKE protocol in the standard model

4.1 Description

Our threshold GPAKE protocol builds upon the PAKE protocol proposed in [15]. The construction is under the common reference string (CRS) model, where all the parties have access to the public parameters that are drawn from a predetermined distribution. Let p, q be two large primes such that $q|(p-1)$; Let $\mathbb{G}_q$ be the multiplicative subgroup of F_p^* of order q; g, h are uniformly random generators of $\mathbb{G}_q$; H is a collision resistant hash function; e is the public key of a CCA secure public key encryption scheme E (we should note that the secret key of E_e is unknown to anyone); Let $UH : \mathbb{G}_q \to \{0,1\}^{3l}$ be a universal hash function [37], where l is the security parameter. The protocol has two phases: initialization phase and session key exchange phase.

Initialization phase. In this phase, a client chooses a password pw uniformly from the dictionary $\mathcal{D}$. We assume that the password pw can be mapped to a member of Z_q^*. For the remainder of the paper, we use passwords as if they were elements of Z_q^*. n servers share the password pw via a distributed Pederson VSS scheme. More precisely, the client splits the password pw into n random values $pw_1, \ldots, pw_n$ such that $pw = pw_1 + \cdots + pw_n \bmod q$. The value pw_i is given to server S_i from a private channel. Each server S_i shares the value pw_i using a (t, n) Pederson VSS scheme. We denote this process by $\mathrm{PedVSS}(pw_i, pw_i')[g,h] \xrightarrow{f_{1,i}, d_{1,i}} (pw_{i,j}, pw_{i,j}')(EP_{i_0}, \ldots, EP_{i_t})$. Eventually, each server S_i sums up all the sub-shares received from the threshold Pederson VSS schemes and obtains the shares $p_i = \Sigma_{j=1}^n pw_{j,i}, p_i' = \Sigma_{j=1}^n pw_{j,i}'$. The servers also compute $EP_i = \prod_{j=0}^n EP_{j_i}$. The verification commitment for (p_i, p_i') is $VP_i = \prod_{k=0}^t (EP_k)^{i^k}$. Each server S_i should save the whole transcript T_p of the distributed Pederson VSS for future use. We should note that all this work should be performed locally by a trusted process.

Session key exchange phase. Assume a client wants to establish a secure session key with the gateway via the help of the servers. The protocol proceeds as follows:

(1) The client randomly chooses a private number $x \in Z_q^*$, computes an ElGalmal ciphertext $X_1||X_2$ of the password, where $X_1 = g^x$ and $X_2 = h^x g^{-pw}$. The client sends (C, X_1, X_2) to the gateway, where C is its identity.

(2) Upon receiving a message from the client, the gateway chooses a random element $y \in Z_q^*$ and computes $Y = g^y$, then sends the message (C, X_1, X_2, Y) to the combiner. Without loss of generality, we assume S_1 is the combiner. The combiner S_1 broadcasts the message (C, X_1, X_2, Y) to all the servers.

(3) Upon receiving the message from the combiner, the servers jointly perform the following steps.

(a) The servers jointly share two random numbers $u_1, u_2 \in Z_q^*$ via a distributed Pederson VSS scheme: $\mathrm{RandPedVSS}([u_1],[u_2])[g,h] \xrightarrow{f_2, d_2} (u_{1,i}, u_{2,i}, T_u)(EU_0, \ldots, EU_t)$.

The verification commitment for $u_{1,i}, u_{2,i}$ is denoted by $VU_i = \prod_{k=0}^t (EU_k)^{i^k}$.

(b) The servers jointly compute a new sharing of the product $u_2 \cdot pw$ using the transcripts for secrets u_2 and pw by the commitment multiplication proof (CMP) protocol of Abe et al. [38]. More precisely, each server performs the following steps.

• Each server S_i randomly picks t-degree polynomials $f_{3,i}$, $d_{3,i}$ and $d_{4,i}$ from $Z_q[X]$ such that $f_{3,i}(0) = u_{2,i}$ and $d_{3,i}(0) = u_{1,i}$. Let $r_{4,i}^0$ denote a randomly chosen free term of $d_{4,i}$, i.e. $d_{4,i}(0) = r_{4,i}^0$. S_i shares $u_{2,i}$ twice as

$$\mathrm{PedVSS}(u_{2,i}, u_{1,i})[h,g] \xrightarrow{f_{3,i}, d_{3,i}} (u_{2,i,j}, u_{1,i,j})(\langle EU1_{i_0} \rangle, \ldots, EU1_{i_t}),$$

$$\mathrm{PedVSS}(u_{2,i}, r_{4,i}^0)[VP_i, h] \xrightarrow{f_{3,i}, d_{4,i}} (u_{2,i,j}, r_{4,i,j}^0)(EU2_{i_0}, \ldots, EU2_{i_t}).$$

Note that VP_i denotes the verification commitment for (p_i, p_i'). Server S_i then randomly selects two t-degree polynomials $f_{5,i}$, $d_{5,i}$ that satisfy $f_{5,i}(0) = u_{2,i} \cdot pw_i \bmod q$ and $d_{5,i}(0) = pw_i' \cdot u_{2,i} + r_{4,i}^0 \bmod q$, and performs

$$\mathrm{PedVSS}(u_{2,i} \cdot pw_i, pw_i' \cdot u_{2,i} + r_{4,i}^0)[g,h] \xrightarrow{f_{5,i}, d_{5,i}} (pu_{2,i,j}, pu_{1,i,j})(\langle EPU_{i_0} \rangle, \ldots, EPU_{i_t}).$$

Angle brackets mean that the value can be locally computed by the servers, or it has been sent before. Note that $EU1_{i_0} = g^{u_{1,i}} h^{u_{2,i}}$ can be computed from the transcript T_u and $EPU_{i_0} = EU2_{i_0}$ is sent before.

• Each server S_j verifies the validity of the shares received from S_i. If server S_j finds that the shares sent from S_i is invalid, S_i is requested to privately send the shares again. Otherwise S_i is disqualified.

• Let I be a set of qualified servers in above step. $|I| \geqslant 2t+1$ must hold. Each server S_j in I computes $pu_j = \sum_{i\in I} \lambda_{i,I} \cdot pu_{2,i,j} \bmod q$, $pu'_j = \sum_{i\in I} \lambda_{i,I} \cdot pu_{1,i,j} \bmod q$, where $\lambda_{i,I}$ is the Lagrange interpolation coefficient. Note that pu_j and pu'_j is a new distributed Pederson VSS sharing of the product $u_2 \cdot pw$. We use T_{pu} to denote the transcript of this process.

(c) Each server S_i also encrypts its local hash key $(u_{1,i}, u_{2,i})$, i.e. $C_i = (E_e(u_{1,i}), E_e(u_{2,i}))$. Define a language L^* as $L^* = \{(C^*, EU^*) : \exists k_1, k_2 \in Z_q^* \text{ s.t. } EU^* = g^{k_1} h^{k_2} \text{ and } C^* = (E_e(k_1), E_e(k_2))\}$. Each server S_i also computes an NIZK proof π_i that $(C_i, VU_i) \in L^*$.

(4) The servers jointly compute the shared exponentiation of secrets:

$$\text{Shared-Exp}[T_u, T_{pu}](X_1, X_2, g) \to (\sigma = X_1^{u_1} X_2^{u_2} g^{u_2 pw}).$$

The combiner S_1 obtains the values $r||\tau_1||\tau_2$ from the universal hash functions UH using σ as the input. The combiner S_1 computes $\sum = H(X_1, X_2, Y, C_i, \pi_i, T_u, C, G, S)$ and encrypts $\omega = E_e[\sum; r]$ using r as the random input, where $i \in I$. Then the combiner S_1 sends $(\omega, C_i, \pi_i, \tau_1, T_u)$ to the gateway, where τ_1 is stored by the gateway to authenticate the client and T_u is the transcript of the Pederson's VSS schemes when sharing u_1 and u_2. Upon receiving the message, the gateway just forwards $(G, \omega, Y, C_i, \pi_i, T_u)$ to the client.

(5) Upon receiving $(G, \omega, Y, C_i, \pi_i, T_u)$, the client first verifies the validity of each π_i. If all the verification is valid, then the client computes $\sigma = EU_0^x = g^{u_1 x} h^{u_2 x}$ and $r'||\tau'_1||\tau'_2 = UH(\sigma)$. Then he verifies whether ω is a ciphertext of $H(X_1, X_2, Y, C_i, \pi_i, T_u, C, G, S)$ using random bits r'. If the verification is successful, then he accepts and outputs the session key $K = Y^x = g^{xy}$. Finally, he sends (C, τ'_1, τ'_2) to the gateway.

(6) Upon receiving (C, τ'_1, τ'_2), the gateway checks whether $\tau'_1 = \tau_1$ or not, if the verification is successful, the gateway also computes the session key $K = X_1^y = g^{xy}$. Finally, the gateway sends (C, τ'_2) to the combiner S_1. The combiner S_1 broadcasts (C, τ'_2) to all the servers. Each server verifies whether τ'_2 is valid or not to detect the password guessing attack.

The session key exchange phase of our protocol is illustrated in Figure 1. We should note that to improve the efficiency of our protocol, u_1 and u_2 are obtained from one distributed Pederson VSS scheme. In this way, we avoid one distributed Pederson VSS execution and one shared exponentiation of secrets execution. However, the security of the shared exponentiation of secrets is still the same. The simulation of the shared exponentiation of secrets in our protocol is exactly the same as the one in [23].

4.2 Security

In this section, we prove the security of our protocol within the security model given in Section 2. As the following theorem states, our protocol is a secure threshold GPAKE protocol as long as the DDH problem is intractable. The proof of Theorem 1 can be found in Appendix A. The security analysis for key privacy and password protection is the same as the one in [14]. We omit it due to lack of space.

Theorem 1. Let $\mathcal{P}$ be the threshold GPAKE protocol in Figure 1. Assume E is a CCA secure public key cryptosystem. Let $\mathcal{A}$ be an adversary which corrupts at most t servers and makes Q_{send}, $Q_{\text{send}} < |D|$, queries of type Send to different instances. Then under the DDH assumption, the adversary's advantage in attacking the semantic security of the proposed protocol is bounded by

$$\text{Adv}^{\text{ake}}_{\mathcal{P},\mathcal{D}}(\mathcal{A}) \leqslant \frac{Q_{\text{send}}}{|\mathcal{D}|} + \text{negl}(l).$$

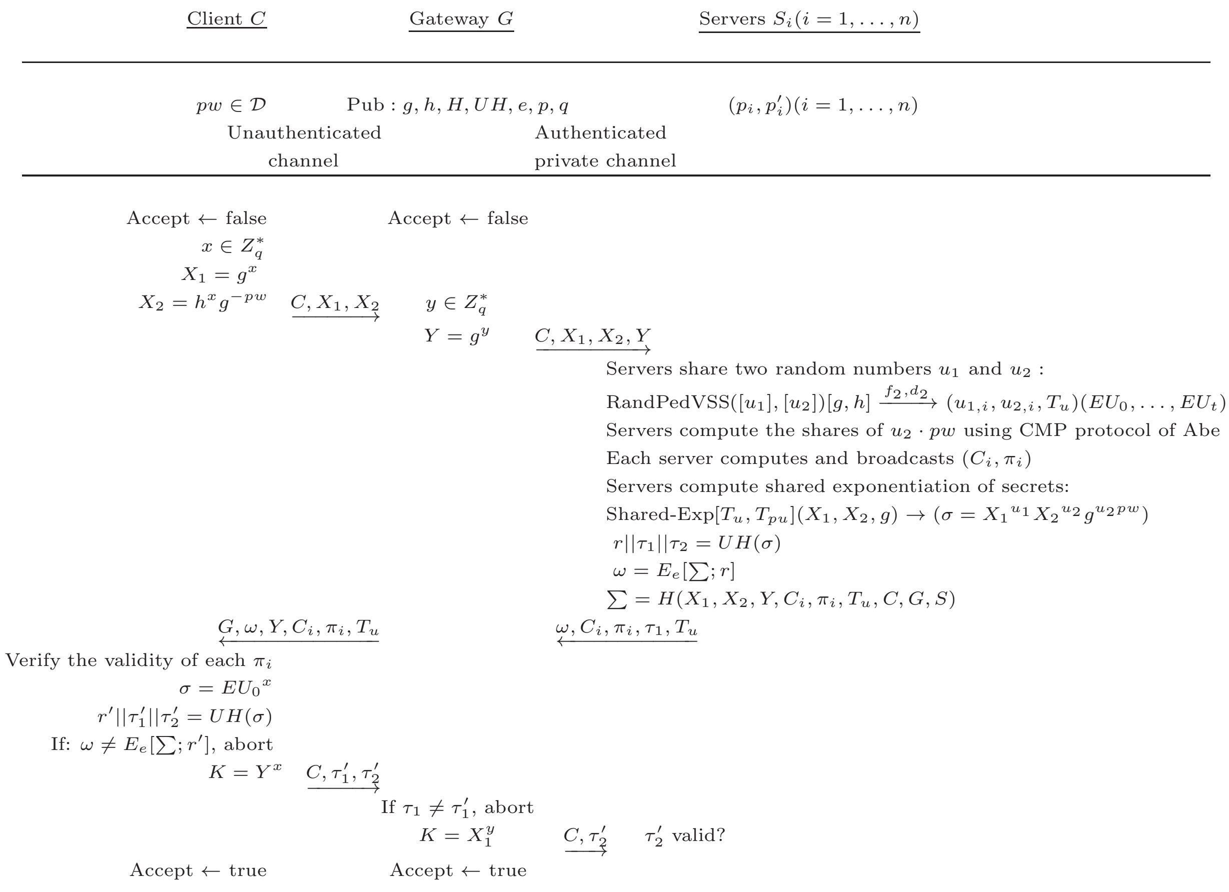
Client C
Gateway G
Servers $S_i (i = 1, \ldots, n)$
$pw \in \mathcal{D}$
Pub : g, h, H, UH, e, p, q
$(p_i, p'_i)(i = 1, \ldots, n)$
Unauthenticated channel
Authenticated private channel
Accept ← false
Accept ← false
$x \in Z_q^*$
$X_1 = g^x$
$X_2 = h^x g^{-pw}$
$\xrightarrow{C, X_1, X_2}$
$y \in Z_q^*$
$Y = g^y$
$\xrightarrow{C, X_1, X_2, Y}$
Servers share two random numbers u_1 and u_2 :
RandPedVSS$([u_1], [u_2])[g, h] \xrightarrow{f_2, d_2} (u_{1,i}, u_{2,i}, T_u)(EU_0, \ldots, EU_t)$
Servers compute the shares of $u_2 \cdot pw$ using CMP protocol of Abe
Each server computes and broadcasts (C_i, π_i)
Servers compute shared exponentiation of secrets:
Shared-Exp$[T_u, T_{pu}](X_1, X_2, g) \to (\sigma = {X_1}^{u_1} {X_2}^{u_2} g^{u_2 pw})$
$r||\tau_1||\tau_2 = UH(\sigma)$
$\omega = E_e[\sum; r]$
$\sum = H(X_1, X_2, Y, C_i, \pi_i, T_u, C, G, S)$
$\xleftarrow{G, \omega, Y, C_i, \pi_i, T_u}$
$\xleftarrow{\omega, C_i, \pi_i, \tau_1, T_u}$
Verify the validity of each π_i
$\sigma = {EU_0}^x$
$r'||\tau'_1||\tau'_2 = UH(\sigma)$
If: $\omega \neq E_e[\sum; r']$, abort
$K = Y^x$
$\xrightarrow{C, \tau'_1, \tau'_2}$
If $\tau_1 \neq \tau'_1$, abort
$K = X_1^y$
$\xrightarrow{C, \tau'_2}$
τ'_2 valid?
Accept ← true
Accept ← true

Figure 1 Threshold GPAKE protocol in the standard model.

5 Extension

The primary idea of our threshold GPAKE protocol is to let the client and n servers execute a threshold PAKE protocol to establish a common secret value. The Diffie-Hellman key exchange materials between the client and the gateway are authenticated using the secret value. Consequently, a threshold PAKE protocol can be derived from our threshold GPAKE protocol if we omit the participation of the gateway. Figure 2 shows the derived threshold PAKE protocol, where $\mathcal{F}$ is a pseudorandom function family and its realization with secret key σ is denoted by $F_\sigma()$. The protocol in Figure 2 can be viewed as a distributed version of Jiang and Gong's protocol [15]. For simplicity, we do not give a formal proof to this protocol. To the best of our knowledge, this protocol is the second threshold PAKE protocol in the standard model (Raimondo and Gennaro's protocol in [23] is the first one). Our protocol is to Jiang and Gong's protocol [15] as the work of Raimondo and Gennaro [23] is to that of Katz-Ostrovsky-Yung [25]. Although the techniques used in our protocol are basically the same as the ones in [23], our protocol is simple and efficient because we try our best to keep the executions of the distributed Pederson VSS to a minimum.

6 Performance analysis

In this section, we compare the efficiency and security properties of the proposed protocols with that of the MSJ06 protocol of MacKenzie et al. [22], the RG06 protocol of Raimondo et al. [24] and the ACFP05 protocol of Abdalla et al. [8]. The performance comparisons are summarized in Table 2. With respect to computation, we consider the number of modular exponentiations. We instantiate the CCA secure cryptosystem E used in our protocol using the DHIES encryption scheme [39]. The encryption cost of

Client C		Servers $S_i(i=1,\ldots,n)$
$pw \in \mathcal{D}$		$(p_i, p'_i)(i=1,\ldots,n)$
$x \in_R Z_q$		
$X_1 = g^x$		
$X_2 = h^x g^{-pw}$	$\xrightarrow{C, X_1, X_2}$	
		Servers share two random numbers u_1 and u_2 :
		RandPedVSS$([u_1],[u_2])[g,h] \xrightarrow{f_2,d_2} (u_{1,i}, u_{2,i}, T_u)(EU_0,\ldots,EU_t)$
		Servers compute the shares of $u_2 \cdot pw$ using the CMP protocol of Abe
		Each server computes (C_i, π_i)
		Servers compute shared exponentiation of secrets:
		Shared-Exp$[T_u, T_{pu}](X_1, X_2, g) \rightarrow (\sigma = {X_1}^{u_1} {X_2}^{u_2} g^{u_2 pw})$
		$r = F_\sigma(3)$
		$\omega = E_e[\sum; r]$
		$\sum = H(X_1, X_2, C_i, \pi_i, T_u, C, S)$
	$\xleftarrow{\omega, C_i, \pi_i, T_u}$	
Verify the validity of each π_i		
$\sigma = {EU_0}^x$		
$r = F_\sigma(3)$		
If: $\omega \neq E_e[\sum; r]$, abort		
$\tau = F_\sigma(2)$		
$sk = F_\sigma(1)$	$\xrightarrow{C, \tau}$	
		If: $\tau \neq F_\sigma(2)$, abort
		$sk = F_\sigma(1)$

Figure 2 Threshold PAKE protocol in the standard model.

the DHIES public key encryption is 2 exponentiations. We also instantiate the non-interactive zero-knowledge proof in our protocol using the relatively-sound non-interactive zero-knowledge proofs in [40]. In terms of communication, we consider the number of message flows exchanged between the client, the gateway and the servers (we do not consider the messages exchanged among the servers). We denote UODA, ROM and PCDDH as the undetectable on-line dictionary attack, the random Oracle model and the password-based chosen-basis decisional Diffie-Hellman assumption [8], respectively. The MSJ06 and the RG06 protocols are two threshold PAKE protocols, the gateway is not involved in these protocols. In addition, undetectable on-line dictionary attacks are usually not considered in TPAKE protocols. We use N/A to denote some operations or security properties that are not considered in these protocols.

With respect to computation cost, we focus on the computation cost of each server. According to [23], the cost of the distributed Pederson VSS scheme for a server is about $2(n+t)$ exponentiations. The cost of shared exponentiation of secrets protocol on l secrets is about ln exponentiations. Finally, the cost of the MCP protocol of Abe et al. [38] for each server is the same as performing three distributed Pederson VSS schemes. Table 2 shows that the computation complexity of our protocol is more efficient than all the other protocols except the one by Abdalla et al. [8]. However, most of the exponentiations can be performed off-line. For example, if $t = 2$ and $n = 6$, each server only needs to compute $3n + 8$ exponentiations on-line in our protocol. Consequently, our protocol is very efficient if the servers save the values computed in the off-line phase. It is also worth mentioning that the computation cost of our protocol is about half the cost of the RG06 protocol.

With respect to computation and communication costs, the only protocol that outperforms our protocol is the ACFP05 protocol. In the number of flows, Abdalla's protocol [8] only needs 4 messages, which is more efficient than our protocol. However, the ACFP05 protocol only achieves implicit authentication and is vulnerable to undetectable on-line dictionary attacks. Although our protocol needs 6 messages, it achieves mutual authentication and is proven to be secure against undetectable on-line dictionary attacks. The security reduction of [8] is performed using the random oracle model. Actually, the computation and communication costs of a protocol in the standard model are usually 3 times to that of its counterpart in the random oracle model. Considering that our protocol is proven secure in the standard model, the computation and communication costs are acceptable. What's more, our protocol does not require

Table 2 Comparisons of efficiency and security

Protocol	MSJ06 [22]	RG06 [24]	ACFP05 [8]	Our protocol 1	Our protocol 2
Exponentiations of client	$16 + t$	15	4	16	15
Exponentiations of gateway	N/A	N/A	2	2	N/A
On-line exponentiations of server	$52 + 38t$	$11n + 6t$	$8 + 4t$	$3n + 8$	$3n + 8$
Off-line exponentiations of server	N/A	$14n + 10t$	N/A	$8(n + t)$	$8(n + t)$
Messages	3	3	4	6	3
Resistance to UODA	N/A	N/A	N	Y	N/A
Security	ROM	Standard	ROM	Standard	Standard
Assumptions	DDH	DDH	DDH, PCDDH	DDH	DDH
Server public keys	Y	N	Y	N	N

the servers' public key to be known to the client. Hence, our protocol is simpler and more practical. Another advantage of our protocol is that the proof of its security relies on the simple and standard DDH assumption, while the security of ACFP05 also relies on the PCDDH assumption, which is not as standard as the DDH assumption.

7 Conclusion

In this paper, we investigate the design of a threshold GPAKE protocol in the standard model. By extending the 2-party PAKE protocol by Jiang and Gong to the distributed setting, we propose an efficient and simple threshold GPAKE protocol in the common reference string model. The proposed protocol is proven secure under the DDH assumption. In addition, we also present an efficient provably secure threshold PAKE protocol in the standard model. Compared with related protocols, the proposed protocols achieve stronger security with reasonable computation overhead. As a result, our protocols are more suitable for practical applications.

Acknowledgements This work was funded by National High Technology Research and Development Program of China (863 Program) (Grant No. 2015AA016007), National Natural Science Foundation of China (Grant Nos. 61309016, 61379150, 61501515), Postdoctoral Science Foundation of China (Grant No. 2014M562493), Postdoctoral Science Foundation of Shanxi Province, Fundamental Research Funds for the Central Universities (Grant No. JB161501), and Key Scientific Technological Project of Henan Province (Grant Nos. 122102210126, 092101210502).

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Xia Z H, Wang X H, Sun X M, et al. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans Parallel Distrib Syst, 2015, 27: 340–352
2 Fu Z J, Sun X M, Liu Q, et al. Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Trans Commun, 2015, 98: 190–200
3 Ren Y J, Shen J, Wang J, et al. Mutual verifiable provable data auditing in public cloud storage. J Internet Tech, 2015, 16: 317–323
4 Ni L, Chen G L, Li J H, et al. Strongly secure identity-based authenticated key agreement protocols in the escrow mode. Sci China Inf Sci, 2013, 56: 082113
5 He D B, Zeadally S, Xu B W, et al. An efficient identity-based conditional privacy-preserving authentication scheme for vehicular Ad-hoc networks. IEEE Trans Inf Foren Sec, 2015, 10: 2681–2691
6 Wang S B, Zhu Y, Ma D, et al. Lattice-based key exchange on small integer solution problem. Sci China Inf Sci, 2014, 57: 112111
7 He D B, Zeadally S. Authentication protocol for an ambient assisted living system. IEEE Commun Mag, 2015, 53: 71–77
8 Abdalla M, Chevassut O, Fouque P A, et al. A simple threshold authenticated key exchange from short secrets. In: Advances in Cryptology — ASIACRYPT 2005. Berlin: Springer, 2005. 566–584
9 Byun J W, Lee D H, Lim J I. Security analysis and improvement of a gateway-oriented password-based authenticated key exchange protocol. IEEE Commun Lett, 2006, 10: 683–685

10 Kyung S. Cryptanalysis and enhancement of modified gateway-oriented password-based authenticated key exchange protocol. IEICE Trans Fund Electron Commun Comput Sci, 2008, 91: 3837–3839

11 Abdalla M, Izabachene M, Pointcheval D. Anonymous and transparent gateway-based password-authenticated key exchange. In: Cryptology and Network Security. Berlin: Springer, 2008. 133–148

12 Chor B, Kushilevitz E, Goldreich O, et al. Private information retrieval. J ACM, 1998, 45: 965–981

13 Yoon E J, Yoo K Y. An optimized gateway-oriented password-based authenticated key exchange protocol. IEICE Trans Fund Electron Commun Comput Sci, 2010, 93: 850–853

14 Wei F S, Zhang Z F, Ma C G. Gateway-oriented password-authenticated key exchange protocol in the standard model. J Syst Softw, 2012, 85: 760–768

15 Jiang S Q, Gong G. Password based key exchange with mutual authentication. In: Selected Areas in Cryptography. Berlin: Springer, 2005. 267–279

16 Wei F S, Zhang Z F, Ma C G. Analysis and enhancement of an optimized gateway-oriented password-based authenticated key exchange protocol. IEICE Trans Fund Electron Commun Comput Sci, 2013, 96: 1864–1871

17 Chien H Y, Wu T C, Yeh M K. Provably secure gateway-oriented password-based authenticated key exchange protocol resistant to password guessing attacks. J Inf Sci Eng, 2013, 29: 249–265

18 Choi S B, Yoon E J. Cryptanalysis of provably secure gateway-oriented password-based authenticated key exchange protocol. Appl Math Sci, 2013, 7: 6319–6328

19 Ford W, Kaliski B S. Server-assisted generation of a strong secret from a password. In: Proceedings of IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Gaithersburg, 2000. 176–180

20 Jablon D P. Password authentication using multiple servers. In: Topics in Cryptology — CT-RSA 2001. Berlin: Springer, 2001. 344–360

21 MacKenzie P, Shrimpton T, Jakobsson M. Threshold password-authenticated key exchange. In: Advances in Cryptology — CRYPTO 2002. Berlin: Springer, 2002. 385–400

22 MacKenzie P, Shrimpton T, Jakobsson M. Threshold password-authenticated key exchange. J Cryptol, 2006, 19: 27–66

23 Raimondo M, Gennaro R. Provably secure threshold password-authenticated key exchange. In: Advances in Cryptology — EUROCRYPT 2003. Berlin: Springer, 2003. 507–523

24 Raimondo M, Gennaro R. Provably secure threshold password-authenticated key exchange. J Comput Syst Sci, 2006, 72: 978–1001

25 Katz J, Ostrovsky R, Yung M. Efficient and secure authenticated key exchange using weak passwords. J ACM, 2009, 57: 3

26 Lee S, Han K, Kang S, et al. Threshold password-based authentication using bilinear pairings. In: Public Key Infrastructure. Berlin: Springer, 2004. 350–363

27 Chai Z, Cao Z, Lu R. Threshold password authentication against guessing attacks in Ad hoc networks. Ad Hoc Netw, 2007, 5: 1046–1054

28 Li C T, Chu Y P. Cryptanalysis of threshold password authentication against guessing attacks in ad hoc networks. Int J Netw Secur, 2009, 8: 166–168

29 Guo P, Wang J, Li B, et al. A variable threshold-value authentication architecture for wireless mesh networks. J Int Tech, 2014, 15: 929–936

30 Bagherzandi A, Jarecki S, Saxena N, et al. Password-protected secret sharing. In: Proceedings of the 18th ACM Conference on Computer and Communications Security. New York: ACM, 2011. 433–444

31 Jarecki S, Kiayias A, Krawczyk H. Round-optimal password-protected secret sharing and t-pake in the password-only model. In: Advances in Cryptology — ASIACRYPT 2014. Berlin: Springer, 2014. 233–253

32 Camenisch J, Lehmann A, Lysyanskaya A, et al. Memento: how to reconstruct your secrets from a single password in a hostile environment. In: Advances in Cryptology — CRYPTO 2014. Berlin: Springer, 2014. 256–275

33 Hasegawa S, Isobe S, Iwazaki J Y, et al. A strengthened security notion for password-protected secret sharing schemes. IEICE Trans Fund Electron Commun Comput Sci, 2015, 98: 203–212

34 Chaum D, Pedersen T P. Wallet databases with observers. In: Advances in Cryptology — CRYPTO'92. Berlin: Springer, 1993. 89–105

35 Katz J, Vaikuntanathan V. Round-optimal password-based authenticated key exchange. J Cryptol, 2013, 26: 714–743

36 Pedersen T P. Non-interactive and information-theoretic secure verifiable secret sharing. In: Advances in Cryptology — CRYPTO'91. Berlin: Springer, 1992. 129–140

37 Hastad J, Impagliazzo R, Levin L A, et al. A pseudorandom generator from any one-way function. SIAM J Comput, 1999, 28: 1364–1396

38 Abe M, Cramer R, Fehr S. Non-interactive distributed-verifier proofs and proving relations among commitments. In: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology. London: Springer, 2002. 206–223

39 Abdalla M, Bellare M, Rogaway P. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Proceedings of the Conference on Topics in Cryptology: the Cryptographer's Track at RSA. London: Springer, 2001. 143–158

40 Jutla C, Roy A. Relatively-sound NIZKs and password-based key-exchange. In: Proceedings of the 15th International Conference on Practice and Theory in Public Key Cryptography. Berlin: Springer, 2012. 485–503

Appendix A Proof of Theorem 1

Proof. We prove Theorem 1 through a sequence of hybrid experiments. We will prove that the advantage of an adversary $\mathcal{A}$ in distinguishing the views between a system using the real password pw and a system using a dummy password $\widetilde{pw}$ is at most $\frac{Q_{\rm send}}{|\mathcal{D}|}$. The view of $\mathcal{A}$ is the collection of messages that $\mathcal{A}$ sees during the execution of the protocol. Since we use information-theoretic secure Pederson VSS scheme, we can ignore the public information of VSS schemes in the simulation. In respect of the view of $\mathcal{A}$, we focus on the public information that the protocol reveals. More specifically, the view of the adversary $\mathcal{A}$ when the password pw is used includes the following values: $(X_1 = g^x,\ X_2 = h^x g^{pw},\ \tau_1',\ \tau_2')$ generated by the client, $Y = g^y$ generated by the gateway, and $(\omega, C_i, \pi_i, \tau_1, T_u)$ generated by the servers. The view of the adversary also includes $\sigma = {X_1}^{u_1} {X_2}^{u_2} g^{u_2 pw}$ computed on the servers's side. We denote the distribution of these values with view(pw). In the following, we will prove the views view(pw) and $\widetilde{pw}$ are indistinguishable in passive session under the DDH assumption and the views view(pw) and $\widetilde{pw}$ can be distinguished with probability at most $\frac{Q_{\rm send}}{|\mathcal{D}|}$.

We use P_i, Adv $(\mathcal{A}, P_i)$ to denote the i-th hybrid experiment and the advantage of $\mathcal{A}$ when participating in experiment P_i, respectively. Given a password pw of a client, we say the ciphertext $X_1||X_2$ is consistent if $\log_g X_1 = \log_h(X_2 g^{pw})$. Otherwise, the ciphertext is inconsistent.

Experiment P_0. This experiment is the real attack game defined in the security model. According to the definition in Section 2, the adversary $\mathcal{A}$ can make a number of oracle calls (Send, Reveal, Execute and Test). It is obvious that

$$\mathrm{Adv}\,(\mathcal{A}) = \mathrm{Adv}\,(\mathcal{A}, P_0)\,.$$

Experiment P_1. In this experiment, we begin to change the simulation rules to Execute queries. In response to a query Execute$(C^i, G^j, (S_{n1}^{l_{n1}}, \ldots, S_{nk}^{l_{nk}}))$, we now simulate the Execute queries with a dummy password $\widetilde{pw}$. We simulate the client with messages $X_1 = g^x$, $X_2 = h^x g^{\widetilde{pw}}$. We simulate the rest of the protocol according to the description of the protocol except that we choose σ randomly from $\mathbb{G}_q$ and simulate the protocol Shared-Exp using the randomly chosen value σ. Note that X_1 and X_2 is inconsistent with respect to the real password pw. As is proven in [8], if $X_1||X_2$ is inconsistent, then the value σ is uniformly random in $\mathbb{G}_q$. The view of the adversary in this experiment is indistinguishable from the view in previous experiment. Assume that the adversary $\mathcal{A}$ can distinguish between view(pw) and view$(\widetilde{pw})$, we show how to break the semantic security of the ElGamal encryption using $\mathcal{A}$ as a subroutine in the following.

We are given an instance of the ElGamal encryption scheme where g is a generater of $\mathbb{G}_q$ and h is the public key. We are also given an ElGamal ciphertext (X_1, X_2) which is either an encryption of g^{pw} or an encryption of $g^{\widetilde{pw}}$. More precisely, $X_1 = g^x$ and $X_2 = h^x g^{pw}$ or $X_2 = h^x g^{\widetilde{pw}}$. We build a simulated view as follows. We first simulate the client with the message X_1, X_2. We also simulate the gateway with $Y = g^y$, where $y \in_R Z_q$. We then run the servers, letting the adversary control t of them. Firstly, we execute the distributed Pederson VSS scheme to generate u_1 and u_2. Then we simulate the CMP protocol of Abe as if the ciphertext (X_1, X_2) is an encryption of g^{pw} and the correct password is pw. We finally simulate the Shared-Exp using the value $\sigma = {X_1}^{u_1}(X_2/g^{pw})^{u_2}$. The σ value is also used in the client side. Note that if $X_2 = h^x g^{pw}$, the above simulation follows the distribution of view(pw). Otherwise if $X_2 = h^x g^{\widetilde{pw}}$, the simulation follows the distribution of view$(\widehat{pw})$, where the dummy password $\widehat{pw} = pw - \widetilde{pw}$. Thus if the adversary can distinguish the views between the current experiment and the previous one, we can break the semantic security of the ElGamal encryption scheme. We have

$$|\mathrm{Adv}\,(\mathcal{A}, P_1) - \mathrm{Adv}\,(\mathcal{A}, P_0)| \leqslant \mathrm{negl}(n).$$

Experiment P_2. In this experiment, we again change the simulation of the Execute oracle so that the session key K is chosen uniformly at random from $\mathbb{G}_q$. The difference in the advantage between the current experiment and previous one is at most that of breaking the DDH assumption.

To prove this, let us be given an instance of the DDH problem (U, V, W). To simulate the Execute(C^i, G^j) oracle, we first choose a_0, a_1, b_0, b_1 in Z_q. Then, we compute $X_1 = U^{a_0} g^{a_1}$ and $Y = V^{b_0} g^{b_1}$. All other simulation is the same as the one in the experiment P_1. Finally, we set the Diffie-Hellman key K to $W^{a_0 b_0} \cdot U^{a_0 b_1} \cdot V^{a_1 b_0} \cdot g^{a_1 b_1}$. If (U, V, W) is a true Diffie-Hellman triple, then the simulation is perfectly identical to the one in P_1; On the other hand, when (U, V, W) is a random triple, then the simulation is the one in P_2. If an adversary can distinguish between the experiments P_1 and P_2, then he can solve the DDH problem with exactly the same advantage. Hence we have

$$|\mathrm{Adv}\,(\mathcal{A}, P_2) - \mathrm{Adv}\,(\mathcal{A}, P_1)| \leqslant \mathrm{negl}(n).$$

Note that Execute queries in P_2 generate random session keys and we use the dummy password $\widetilde{pw}$ that are independent to the actual password in the simulation of the client. Although we still use the real password in the simulation of the servers, no password information is revealed to the adversary due to the simulation soundness of the CPM protocol and the Shared-Exp protocol. As a result, the adversary's advantage from passive sessions is negligible in P_2.

Experiment P_3. In this experiment, we will modify the simulation rules for the Send queries. For convenience, we use Send$_0(C^i, G^j)$ to denote a message that causes the client instance C^i to initiate the protocol with gateway G; We use Send$_1(G^j, (C, X_1, X_2))$ to denote the first message sending to the gateway instance G^j. The channel between the gateway and the combiner is assumed to be an authenticated private channel, so the adversary cannot change the messages send in this channel. However, the adversary can eavesdrop the messages transmitted in this channel since it controls up to t corrupted servers. Let Send$_2(C^i, (G, \omega, Y, C_i, \pi_i, T_U))$ denote sending the message to client instance C^i. Let Send$_3(G^j, (C, \tau_1', \tau_2'))$ denote sending the message to gateway instance G^j.

From this experiment on, we change the way public parameters are generated. More precisely, we choose $s \in Z_q$ and set $h = g^s$. We also record the secret key d of the cryptosystem E when the public key e in CRS is generated. These trapdoor information can be used to determine whether the adversary guesses the correct password or not. Suppose a gateway instance G^j receives a $\text{Send}_1(G^j, (C, X_1, X_2))$ message, if (X_1, X_2) is generated by the adversary himself (not a replay of the client's message), the simulator decrypts (X_1, X_2) using the secret key s to see if it is a valid encryption of the password. If (X_1, X_2) is consistent, which means the adversary guesses the correct password, we terminate the simulation and declare the adversary wins the attack game. This introduces a new way for the adversary to win the attack game, so it is clear this only increase the success probability of the adversary:

$$\text{Adv}\,(\mathcal{A}, P_2) \leqslant \text{Adv}\,(\mathcal{A}, P_3)\,.$$

Experiment P_4**.** In this experiment, we continue to modify the simulation rules for Send queries. For a $\text{Send}_1(G^j, (C, X_1, X_2))$ query, we first decrypt the ciphertext (X_1, X_2) to see if it is a valid encryption of the password. If it is valid, we declare that the adversary succeeds. If it is invalid, we change the simulation rules for the servers. More precisely, we let the honest servers randomly choose the value σ and simulate the protocol Shared-Exp using σ. Note that the adversary is allowed to corrupt at most t servers, so the above simulation is perfect. Moreover, when the adversary sends a $\text{Send}_3(G^j, (C, \tau_1', \tau_2'))$ query, we simply reject and terminate the session without checking the validity of τ_1' and τ_2'. All other simulation rules are same as the previous experiment.

Since the ciphertext (X_1, X_2) is inconsistent, then the value σ is uniformly random in $\mathbb{G}_q$. This experiment and the previous one is indistinguishable unless the adversary guesses the randomly chosen value σ or the adversary simply guesses the correct values τ_1' and τ_2'. However, the probabilities of above events are negligible. Thus, we have

$$|\text{Adv}\,(\mathcal{A}, P_4) - \text{Adv}\,(\mathcal{A}, P_3)| \leqslant \text{negl}(n).$$

Experiment P_5**.** In this experiment, we change the simulation rules for the Send_0 and Send_2 queries. When the adversary asks a $\text{Send}_0(C^i, G^j)$ query, we randomly choose X_1, X_2 from $\mathbb{G}_q$. Moreover, when the adversary sends back a $\text{Send}_2(C^i, (G, \omega, Y, C_i, \pi_i, T_U)$ query, we use the trapdoor information generated in Experiment P_3 to check whether the adversary guesses the correct password or not. More precisely, we first verify the correctness of each zero-knowledge proof π_i. If all the proofs are valid, we decrypt each C_i using the secret key d to get the shares $u_{1,i}$ and $u_{2,i}$. Then we can reconstruct the secret values u_1 and u_2. Now we can compute the secret value $\sigma = {X_1}^{u_1} {X_2}^{u_2} g^{u_2 pw}$ using u_1, u_2 and the real password pw. We then compute the randomness r by the universal hash function using σ as input. Finally, we verify if ω is an encryption of $\sum$ with r as its random input, where $\sum = H(X_1, X_2, Y, C_i, \pi_i, T_u, C, G, S)$. If the verification of ω is successful, then we terminate the simulation and declare the adversary succeeds in the attack game. If ω is invalid, we let the client instance terminates without accepting. Note that the adversary only controls at most t servers, thus the above simulation is perfect. With a similar analysis to the experiment P_3, we have

$$\text{Adv}\,(\mathcal{A}, P_4) \leqslant \text{Adv}\,(\mathcal{A}, P_5)\,.$$

Experiment P_6**.** In this experiment, we consider the case in which the adversary relays the messages from the oracle instances using Send queries. In other words, the adversary performs passive attacks using the Send oracle. The simulation rules for this scenario is exactly the same as passive sessions. With a similar analysis to the experiments P_2 and P_3, we have

$$|\text{Adv}\,(\mathcal{A}, P_6) - \text{Adv}\,(\mathcal{A}, P_5)| \leqslant \text{negl}(n).$$

In this final experiment, all the passive session keys are chosen uniformly random and we use a dummy password in the simulation of the client instances. Although we still use the real password in the simulation of servers, this will leak any information of the password because the distributed Pederson VSS scheme is theoretic secure. With respect to active sessions, we terminate the simulation and declare the adversary succeeds if the adversary correctly guesses the password. All the active sessions in which the adversary guesses a wrong password are terminate without accepting. As a result, the advantage of adversary in distinguishing the session key is negligible.

In the final experiment, the adversary can win the attack game in the following ways: (a) the adversary correctly guesses the real password and sends a valid $\text{Send}_1(G^j, (C, X_1, X_2))$ query or a valid $\text{Send}_3(G^j, (C, \tau_1', \tau_2'))$; (b) the adversary correctly guesses the random bit used by the Test oracle. We use PwdGuess to denote the case (a) occurs, which means the adversary guesses the correct password. Because no information of the password is leaked in the simulation, the success probability of the adversary in winning the attack game is

$$\begin{aligned}
\Pr[\text{Succ}] &\leqslant \Pr[\text{Succ} \wedge \text{PwdGuess}] + \Pr[\text{Succ} \wedge \overline{\text{PwdGuess}}] \\
&\leqslant \Pr[\text{PwdGuess}] + \Pr[\text{Succ}|\overline{\text{PwdGuess}}] \cdot (1 - \Pr[\text{PwdGuess}]) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \Pr[\text{PwdGuess}] \\
&\leqslant \frac{1}{2} + \frac{1}{2} \cdot \frac{Q_{\text{send}}}{|\mathcal{D}|}.
\end{aligned}$$

According to the definition of the advantage of the adversary, Theorem 1 is proven.