SCIS 2015 The 32nd Symposium on
Cryptography and Information Security
Kokura, Japan, Jan. 20 - 23, 2015
The Institute of Electronics,
Information and Communication Engineers

Threshold Two-Move Password Authenticated Key Exchange Protocol

Tatsuya Kyogoku * Minseon Lee * Masayuki Abe †* Tatsuaki Okamoto †*

Abstract: Distributed smooth projective hash functions ($SPHF^x$) allow us to distribute the computation of the hash value across n parties and can be used as building blocks in threshold and multi-party protocols. In this paper we present a new threshold password authenticated key exchange (threshold PAKE) protocol using $SPHF^x$ on Cramer-Shoup Ciphertexts. Our protocol is two-move where a client communicates twice (send and receive) with each server, while the existing threshold PAKE protocols are three-move.

Keywords: password authenticated key exchange, smooth projective hash functions, Pedersen-VSS

1 Introduction

In *password-authenticated key exchange* (PAKE) [1] protocols, two parties (the client and the server) are able to establish one common random value (a session key), used to encrypt and authenticate messages between them, as long as the two parties share a common secret password. Ford and Kaliski [3] proposed *threshold password-authenticated key exchange* (threshold PAKE) in which the password of the client is stored in a shared form among n authentication servers in order to protect the password against adversaries who are able to breach the authentication server. Their protocol is an n-out-of-n scheme, i.e., the password is shared across n servers and all n servers must cooperate to reconstruct the password. As protocols of threshold PAKE that satisfy both t-out-of-n and provably secure in the standard model, Raimondo and Gennaro [8] presented some protocols where at least $t+1$ servers of them must cooperate to learn the password. One of them is called *non-transparent*: the client is aware of how many servers are involved, and at the end of an execution of the protocol the client produces n separate session keys (the client shares one key with each server).

Another important notion is *smooth projective hash functions* (SPHF)s which have been used as a building block for various cryptographic applications, in particular for PAKE. Cramer and Shoup [2] first introduced the concept of SPHF to construct CCA-secure public key encryption schemes. It was then used in the construction of a PAKE scheme by Gennaro and Lindell [4]. KOY scheme [6] is based on their work. Kiefer and Manulis, in their recent paper [5], introduced the notion of (distributed) *extended smooth projective hash functions* (SPHF^x) that allows one to distribute the computation of the hash value across n parties for special languages called divergent parameterized languages. Further they clarified the notion with a concrete instantiation using Cramer-Shoup encryption and proposed a two-server PAKE framework as application of distributed SPHF^x. While they showed the "possibility" that distributed SPHF^x can be used as a building block in threshold PAKE schemes (n servers), they only focused on two-server PAKE framework in their paper.

Our contributions: In this paper, we present a new threshold PAKE scheme using distributed SPHF^x. Our scheme is two-move, i.e., the client communicates twice (send and receive) with each server, while the existing threshold schemes [7, 8] are three-move. Our starting point is the distributed Cramer-Shoup SPHF^x of [5]. For simplicity we call the single server version of the distributed Cramer-Shoup SPHF^x by the centralized Cramer-Shoup SPHF. We transform the centralized Cramer-Shoup SPHF of [5] into a t-out-of-n threshold PAKE scheme. The basic idea of our construction is to distribute the password across n servers using a secret sharing scheme such as Pedersen's one. Then the servers cooperate to compute the messages between the client and the servers. In this scheme, the client basically run n copies of the centralized Cramer-Shoup SPHF, one between the client and each server. Security is proven by a reduction to the security of the underlying centralized Cramer-Shoup SPHF, i.e., we show that if there is an adversary breaks our threshold scheme, then we can construct another adversary who breaks an instance of the centralized Cramer-Shoup SPHF.

Organization of the paper: This paper is organized as follows. In Section 2, we recall the notions of distributed SPHF^x and Threshold PAKE, and present

* Kyoto University
† NTT Secure Platform Laboltories, NTT Corporation

some useful definitions and notations. In Section 3, we give a new threshold PAKE scheme using distributed SPHF^x. Finally we analyze the security of our scheme.

2 Preliminaries

In this section we introduce some technics that we use in our scheme.

2.1 Language Representation

Languages are proposed in [9] while SPHF for languages of chipertexts [10] need some particular cases only. Therefore, in this paper we focus only on as needed. A language L_{aux} is indexed by a parameter aux, consisting of two parts(crs, aux'):the public parts common reference string crs, and private part aux'. More concretely crs contains the global parameters and a public key of an encryption scheme while aux' contains an message that should be encrypted.
For a language L_{aux} we assume there exists a function $\Gamma : Set \rightarrow \mathbb{G}^{k\times n}$, and a family of functions $\Theta : Set \rightarrow \mathbb{G}^{1\times n}$ for positive integers k and n. A cyphertext C is in the language L_{aux} if and only if $\Theta_{aux}(C)$ is a linear combination of (the exponents in) the rows of some matrix Γ.
$(C \in L_{aux}) \Leftrightarrow (\exists\lambda \in \mathbb{Z}_p^{1\times k}$such that $\Theta_{aux}(C) = \lambda \odot \Gamma)$
We use notation$\odot : a \in \mathbb{G}, r \in \mathbb{Z}_p : a \odot r = r \odot a = a^r \in \mathbb{G}$
We furthermore require that a user, who knows witness w of the membership $C \in L_{aux}$, can efficiently compute the above linear combination λ.
A smooth projective hash function for ciphertext language L_{aux} consists of the following four algorithms.

SPHF [10]

-$KGen_H(L_{aux})$
generate a hashing key $k_h \in_R \mathbb{Z}_p^{1\times n}$ for language L_{aux}.

-$KGen_P(k_h, L_{aux}, C)$
derives the projection key $k_p = \Gamma \odot k_h \in \mathbb{G}^{k\times 1}$

-$Hash(k_h, L_{aux}, C)$
outputs the hash value $h = \Theta_{aux}(C) \odot k_h \in \mathbb{G}$.

-$PHash(k_p, L_{aux}, C, w)$
return the hash value h $= \lambda \odot k_p \in \mathbb{G}$, with $\lambda = \Omega(w, C)$ for some $\Omega : 0, 1^* \rightarrow \mathbb{G}^{1\times k}$.

We illustrate variables on CS;

$$\Gamma = \begin{pmatrix} g_1 & 1 & g_2 & h & c \\ 1 & g_1 & 1 & 1 & d \end{pmatrix} \in \mathbb{G}^{2\times 5}$$

$\lambda = (r, r\xi) \in \mathbb{Z}_p^{1\times 2}$for $\Omega(r, C) = (r, r\xi)$
$\Theta_{aux}(C) = (u_1, u_1^{\xi}, u_2, e/g_1^m, v) \in \mathbb{G}^{1\times 5}$
$k_h = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{R} \mathbb{Z}_p^5$

2.2 SPHF^x

We use an extended notion of smooth projective hashing [5] that allows us to distribute the computation of the hash value. $SPHF^x$ can use a set of ciphertexts (C_0,C_1,...,C_x) with specific properties while SPHF can use a single ciphertext. $SPHF^x$ computes two different hash value. The hash value h_0 for C_0 can be either computed with knowledge of the hash key k_{h0} or with the witness $w_1, ..., w_x$ that $C_1, ..., C_x \in L_{aux}$. The hash value h_x can be either computed with knowledge of the hash key $k_{h1}, ..., k_{hx}$ or with the witness w_0 that $C_0 \in L_{aux}$.
Let $(C_0, C_1, ..., C_x) \in L_{aux}$
If There exists $(w_0, w_1, ..., w_x)$ and functions h, g
h(aux') $= (aux'_1, ..., aux'_x)$
$Dec'_{\pi}(C_0) = Dec'_{\pi}(g(C_1, ..., C_x))$
(Dec' denotes modified decryption algorithm π denotes secret key for the corresponding public key pk from crs.) $SPHF^x$ consists of the following six algorithms.

SPHF^x

-$KGen_H(L_{aux})$
generates a hashing key $k_hi \in \mathbb{Z}_p^{1\times n} for i \in \{0, ..., x\}$ and language L_{aux}.

-$KGen_P(k_{hi}, L_{aux})$
derives the projection key $k_{pi} = \Gamma \odot k_{hi} \in \mathbb{G}^{1\times k}$ for $i \in \{0, ..., x\}$.

-$Hash_x(k_{h0}, L_{aux}, C_1, ..., C_x)$
outputs the hash value $h_x = \Theta^x_{aux}(C_1, ..., C_x) \odot k_{h0}$

-$PHash_x(k_{p0}, L_{aux}, C_1, ..., C_x, w_1, ..., w_x)$
returns hash value $h_x = \Pi_{i=1}^x(\lambda^i \odot k_{p0})$
where $\lambda^i = \Omega(w_i, C_i)$.

-$Hash_0(k_{h1}, ..., k_{hx}, L_{aux}, C_0)$
outputs hash value
$h_0 = \Pi_{i=1}^x(\Theta^x_{aux}(C_0) \odot k_h i) = \Theta^x_{aux}(C_0) \odot \Sigma_{i=1}^x k_{hi}$

-$PHash_0(k_{p1}, ..., k_{px}, L_{aux}, C_0, w_0)$
return hash value $h_0 = \Pi_{i=1}^x(\lambda^0 \odot k_{pi})$
where $\lambda^0 = \Omega(w_0, C_0)$

2.3 Cramer-Shoup SPHF^x

We use Labeled Cramer-Shoup Encryption(CS) Scheme throughout this paper . Thus we first recall CS Scheme and in the next place define Cramer-Shoup $SPHF^x$.

CS Scheme

-**Setup** (1^k)
generates a group $\mathbb{G}$ of order p, with a generator g

-**KeyGen()**
$(g_1, g_2) \xleftarrow{R} \mathbb{G}^2$
dk $=(x_1, x_2, y_1, y_2, z) \xleftarrow{R} \mathbb{Z}_p^5$
$c = {g_1}^{x_1}{g_2}^{x_2}, d = {g_1}^{y_1}{g_2}^{y_2}, h = {g_1}^z$
$ek = (g_1, g_2, c, d, h, hk)$.

-Enc$(\ell, ek, M; r)$

message$M \in \mathbb{G}$
random scalar $r = \mathbb{Z}_p$
Collision-resistant hash function family $H \in \mathbb{Z}_p^*$
$h_k \overset{\mathrm{R}}{\leftarrow} H$
$\xi = h_k(\ell, {g_1}^r, {g_2}^r, e)$
ciphertext $C = (\ell, {g_1}^r, {g_2}^r, M \cdot h^r, (cd^{\xi})^r)$
$= (\ell, u_1, u_2, e, v)$

-Dec(ℓ, dk, C)

$\xi = hk(\ell, {g_1}^r, {g_2}^r, e)$
If ${u_1}^{x_1+\xi y_1} \cdot {u_2}^{x_2+\xi y_2} = v$ computes$M = e/u_1^z$
else computes $\perp$

CS SPHF^x

-$k_{hi} \overset{\mathbf{R}}{\leftarrow} KGen_H(L_{aux})$:

$k_{hi} = (\eta_1, \eta_2, \theta, \mu, \nu) \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_p^5$

-$k_{pi} \leftarrow KGen_P(k_{hi}, L_{aux})$:

$$\begin{aligned} k_{pi} &= \Gamma \odot k_{hi} \\ &= \begin{pmatrix} g_1 & 1 & g_2 & h & c \\ 1 & g_1 & 1 & 1 & d \end{pmatrix} \odot (\eta_{1,i}, \eta_{2,i}, \theta_i, \mu_i, \nu_i) \\ &= \begin{pmatrix} g_1^{\eta_{1,i}} & & g_2^{\theta_i} & h^{\mu_i} & c^{\nu_i} \\ & g_1^{\eta_{2,i}} & & d^{\nu_i} & \end{pmatrix} \in \mathbb{G}^{2\times 1} \end{aligned}$$

-$h_x \leftarrow Hash_x(k_{h0}, L_{aux}, C_1, ..., C_x)$:

$$\begin{aligned} h_x &= \Theta^x_{aux}(C_1, ..., C_x) \odot k_{h0} \\ &= (\prod_{i=1}^{x} u_{1,i}, \prod_{i=1}^{x} u_{1,i}^{\xi_i}, \prod_{i=1}^{x} u_{2,i}, \prod_{i=1}^{x} e_i/g_1^m, \prod_{i=1}^{x} v_i) \\ &\quad \odot (\eta_{1,0}, \eta_{2,0}, \theta_0, \mu_0, \nu_0) \\ &= (\prod_{i=1}^{x} u_{1,i})^{\eta_{1,0}} (\prod_{i=1}^{x} u_{1,i}^{\xi_i})^{\eta_{2,0}} (\prod_{i=1}^{x} u_{2,i})^{\theta_0} \\ &\quad (\prod_{i=1}^{x} e_i/g_1^m)^{\mu_0} (\prod_{i=1}^{x} v_i)^{\nu_0} \in \mathbb{G} \end{aligned}$$

-$h_x \leftarrow PHash_x(k_{p0}, L_{aux}, C_1, ..., C_x, r_1, ..., r_x)$:
with $\Omega(r_i, C_i) = (r_i, r_i\xi_i)$

$$\begin{aligned} h_x &= \prod_{i=1}^{x} (\lambda^i \odot k_{p0}) \\ &= \prod_{i=1}^{x} \Bigg((r_i, r_i\xi_i) \\ &\quad \odot \begin{pmatrix} g_1^{\eta_{1,i}} & & g_2^{\theta_i} & h^{\mu_i} & c^{\nu_i} \\ & g_1^{\eta_{2,i}} & & d^{\nu_i} & \end{pmatrix} \Bigg) \\ &= \prod_{i=1}^{x} [(g_1^{\eta_{1,i}} g_2^{\theta_i} h^{\mu_i} c^{\nu_i})^{r_i} (g_1^{\eta_{2,i}} d^{\nu_i})^{r_i\xi_i}] \\ &\quad \in \mathbb{G} \end{aligned}$$

-$h_0 \leftarrow Hash_0(k_{h1}, ..., k_{hx}, L_{aux}, C_0)$:

$$\begin{aligned} h_0 &= \prod_{i=1}^{x} (\Theta^0_{aux}(C_0) \odot k_{hi}) \\ &= \prod_{i=1}^{x} [(u_1, u_1^{\xi}, u_2, e/g_1^m, v) \odot (\eta_{1,i}, \eta_{2,i}, \theta_i, \mu_i, \nu_i)] \\ &= \prod_{i=1}^{x} (u_1^{\eta_{1,i}} u_1^{\xi\eta_{2,i}} u_2^{\theta_i} (e/g_1^m)^{\mu_i} v^{\nu_i}) \in \mathbb{G} \end{aligned}$$

-$h_0 \leftarrow PHash_0(k_{p1}, ..., k_{px}, L_{aux}, C_0, r0)$:

$$\begin{aligned} h_0 &= \prod_{i=1}^{x} (\lambda^0 \odot k_{pi}) \\ &= \prod_{i=1}^{x} \Bigg((r_0, r_0\xi_0) \odot \begin{pmatrix} g_1^{\eta_{1,i}} & & g_2^{\theta_i} & h^{\mu_i} & c^{\nu_i} \\ & g_1^{\eta_{2,i}} & & d^{\nu_i} & \end{pmatrix} \Bigg) \\ &= \prod_{i=1}^{x} [(g_1^{\eta_{1,i}} g_2^{\theta_i} h^{\mu_i} c^{\nu_i})^{r_0} (g_1^{\eta_{2,i}} d^{\nu_i})^{r_i 0 x i_0}] \in \mathbb{G} \end{aligned}$$

2.4 Threshold Scheme

We use Pedersen's Verifiable Secret Sharing(VSS) protocol introduced in [11] as a central tool in our solution. The dealer distributes a secret to n servers such a way that each server can verify that it has received correct share of the secret without interacting with other servers. If qualified servers assemble above threshold level they can find the secret. Else no information about the secret can be obtained. Let math be elements of $\mathbb{G}_q$ such that no one knows $log_g h$. The VSS commits himself to an s $\in \mathbb{Z}_q$ by choosing $t \in \mathbb{Z}_q$ at random and computing $E(s,t) = g^s h^t$. Such a commitment can later be opened by revealing s and t. We distribute n secrets simultaneously with one random value.

Pedersen's VSS

Setup(1^k)

generate $(\mathbb{G}, q, g_1, \cdots, g_n) \overset{\mathrm{R}}{\leftarrow} GenG(1^k)$ and choose h elements of $\mathbb{G}_q$ such that nobody knows $log_{g_1} h, \cdots, log_{g_n} h$.

VSS

When commit messages $m_k (1 \le k \le n) \in \mathbb{Z}_q$

1. Dealer chooses r $\overset{\mathrm{U}}{\leftarrow} \mathbb{Z}_q$.
2. Dealer chooses n polynomials $S_k()$ of degree at most t (satisfying$S_k(0) = m_k$) and let $m_{k,i} = S_k(i) (1 \le i \le n)$.
3. Dealer chooses one polynomials R() of degree at most t at random and let $r_i = R(i)$.
   $E(m_1, \cdots, m_n, r) = g_1^{m_1} \cdots g_n^{m_n} h^r$.
   $E(m_{k,i}, r_i) = g_1^{m_{1,i}} \cdots g_n^{m_{n,i}}$
4. Dealer sends $g_1^{m_1} \cdots g_n^{m_n} h^r$ and coefficients of each polynomial to the servers.

5. Dealer sends $E_i(m_{k,i}, r_i)$ secretly to S_i.

6. Receivers check if c := $g_1^{m_1} \cdots g_n^{m_n} h^r$ is collect.

2.5 Joint sharings

Pedersen-RVSS needs a trusted dealer who generates a shared secret. To avoid the use of the trusted dealer, each server pretend to be a dealer and runs a copy of Pedersens protocol with a random secret. Each random secret is shared among qualified servers and sum of the secrets that we want to share. In this method the sum is taken only over servers that were not disqualified as dealers.
In [12] Joint-Pedersen-RVSS with basis g and h, which generations the shared secret s. It is then followed by a second phase in which each player performs a Feldman's VSS with basis g with the same polynomial used for Joint-Pedersen-RVSS. this second phase will allows the servers to calculate g^s secretly.
We modify this protocol so that it works with n secrets and any combination of basis based on [8]. For simplicity we restrict ourselves to the case of two secrets. It should be clear how to extend it to any number of secrets in the following description. Let $g, h \in \mathbb{G}_q$ be given such that the commitment scheme.the dealer can distribute x, x' $\in \mathbb{Z}_q$ as follows.

Joint-Pedersen-RVSS

1. Each server S_i for {i = 1,$\cdots$,n}performs Pedersen-VSS of random values z_i, z_i' as a dealer.

   (a) Each server S_i Chooses three random polynomials
   $f_i(z), f_i'(z), f_i''(z)$, over $\mathbb{Z}_q$ of degree t.
   $f_i(z) = a_{i0} + a_{i1}z + , \cdots , + a_{it}z^t$
   $f_i'(z) = b_{i0} + b_{i1}z + , \cdots , + b_{it}z^t$
   $f_i''(z) = c_{i0} + c_{i1}z + , \cdots , + c_{it}z^t$
   Let $z_i = a_{i0} = f_i(0)$, $z_i' = b_{i0} = f_i'(0)$, $z_i'' = c_{i0} = f_i''(0)$ S_i and S_i broadcast $V_{ik} = g_1^{a_{ik}} g_2^{b_{ik}} h^{c_{ik}}$ mod p for k = 0,$\cdots$,t.
   S_i computes shares $s_{ij} = f_i(j)$, $s_{ij}' = f_i'(j)$, $s_{ij}'' = f_i''(j)$ mod q for $j = 1, \cdots, n$ and sends s_{ij}, s_{ij}', s_{ij}'' to server S_j.

   (b) Each server S_j verifies the shares it received from other servers. For each i = 1,...,n, S_j checks

   $$g_1^{s_{ij}} g_2^{s'_{ij}} h^{s''_{ij}} = \prod_{k=0}^{t} (V_{ik})^{j^k} \pmod p \quad (1)$$

   If the check fails for an index i, S_j broadcasts a complaint against S_i

   (c) Each server S_i that received a complaint from server S_j broadcasts the values s_{ij}, s_{ij}', s_{ij}'' that satisfy (1).

   (d) Each server marks as disqualified any server that either

   i. received more than t complaints in Step1.(b)
   ii. answered to a complaint in step1.(c) with values that falsify (1)

2. Each server then builds the set of non-disqualified servers QUAL.

3. The distributed secret values x, x' are not explicitly computed by any party, but it equals

   $$x = \sum_{i \in QUAL} z_i \ mod \ q,$$

   $$x' = \sum_{i \in QUAL} z_i' \ mod \ q.$$

   Each server S_i sets his share of the secret as

   $$x_i = \sum_{j \in QUAL} s_{ji} \ mod \ q$$

   $$x_i' = \sum_{j \in QUAL} s_{ji}' \ mod \ q.$$

   $$x_i'' = \sum_{j \in QUAL} s_{ji}'' \ mod \ q.$$

We will use the following notation:

Joint-Pedersen-RVSS(g_1, g_2, h)

$$\xrightarrow[t,n]{f,f',f''} [z_i, z_i', z_i'', T_Z T_{Z'}](V_{ik}, QUAL)\{x, x'\}$$

$T_Z, T_{Z'}$ are transcripts of the n VSS's executed by each server.
T_Z = {private to $S_i (i \in QUAL) : \{a_{ik}\}, \{c_{ik}\}, \{s_{ji}, s_{ji}', s_{ji}''\}$; $public : \{V_{ik}\}$}
$T_{Z'}$ = {private to $S_i (i \in QUAL) : \{b_{ik}\}, \{c_{ik}\}, \{s_{ji}, s_{ji}', s_{ji}''\}$; $public : \{V_{ik}\}$}

Shared-Exp
Input: the transcripts $T_{Z,Z'}$

1. Each server S_i, $i \in$ QUAL, broadcasts $E_{ik} = {g_1}^{a_{ik}} {g_2}^{b_{ik}} h^{c_{ik}}$ mod p for $k \in [0, .., t]$

2. Each server S_i verifies the values broadcast by the other servers in QUAL, for each $i \in$ QUAL, S_j checks

   $$g_1^{s_{ij}} g_2^{s'_{ij}} \stackrel{?}{=} \prod_{k=1}^{t} (E_{ik})^{j^k} mod \ p \quad (2)$$

   If the check fails for an index i, then $S_j$complains against S_i, by revealing in broadcast the values $s_{ij}, s_{ij}', s_{ij}''$. These values satisfy(1) but do not satisfy(2).

3. If S_i receives at least one valid complaint, other servers run the reconstruction phase of S_i' VSS and obtain the values a_{ik}, b_{ik} and E_{ik} for $k = 0, \cdots, t$.

4. Now each server can compute

$$g_1{}^x g_2{}^{x'} = \prod_{i=QUAL} E_{i0} \ mod \ p$$

We will use the following notation :

$$Shared - Exp[T_Z, T_{Z'}](g_1, g_2) \rightarrow (g_1{}^x g_2{}^{x'})$$

3 Our scheme

We propose a scheme that is t-out-of-n threshold version of the Distributed Cramer-Shoup SPHF^x. In this scheme, we assume that the communication between a client and servers is public while the communication among servers are totally connected by a complete private network. The servers cooperate with each other to compute the replies of the jth server for the client in the background (the client is unable to see the server's conversation). We also assume that in the set up phase, the password of the client has previously distributed among the servers in a shared form. The output of the scheme is n separate session keys; the client shares one key with each server.

The main idea of our authentication scheme is to run n copies of the centralized Cramer-Shoup SPHF, one instance between client P_0 and each server S_j. A simple description of the scheme is as follows.

Client P_0 first makes n messages by running n times the first step of the centralized Cramer-Shoup SPHF, and sends the n messages to the n servers, one ($C_{0,j}$, $k_{p0,j}$) for each server S_j.

The servers answer to the client with n independent messages (one sent from each server). For each server S_j, the servers execute Joint-Pedersen-RVSS once to select random values $\eta_{1,j}$, $\eta_{2,j}$, θ_j, μ_j, ν_j, r_j. And using Shared-Exp two times and Joint-Pedersen-RVSS once, they compute $k_{pj} = (g_1{}^{\eta_{1,j}} g_2{}^{\theta_j} h^{\mu_j} c^{\nu_j}, g_1{}^{\eta_{2,j}} d^{\nu_j})$ and $C_j = (\ell_j, g_1{}^{r_j}, g_2{}^{r_j}, h^{r_j} g_1{}^{pw}, (cd^{\xi_j})^{r_j})$, respectively. Then server S_j sends to client P_0 its message (C_j, k_{S_j}).

Lastly, using Shared-Exp once, n servers locally compute n session keys, one key sk_j for each server S_j.

We describe the scheme in more detail in Section3.2, 3.3 and Section 3.4.

3.1 Communication model

We use the same communication model in[8]. The communication between the client and the authentication servers, is carried on a basically insecure network. One the other hand, servers $S_1, \cdots, S_n$ are connected by a complete network of private.

3.2 Set up

We perform set up phase in security that mean QUAL = n.

1. The client P_0 splits his password pw into n random values.
   $pw = pw_1 +, \cdots, +pw_n$
2. The client hands out the values pw_j to the servers $S_j (1 \leq j \leq n)$.
3. Each server performs the following.

$$Joint - Pedersen - RVSS(g_1, h)$$

$$\xrightarrow[t,n]{PW,\tilde{X}} [pw_i, \tilde{x}_i, T_{pw}](V_{pw})\{pw\}$$

3.3 The scheme of the client's side

Public information: $\mathbb{G}$, p, g_1, g_2, g_3, g_4, g_5, g_6, h, c, d, H_k, L_{aux}, pk.

Input for client P_0: the password $pw \in \mathbb{Z}_p$.

1. Client P_0 generates n messages by running n times the first step of the centralized Cramer-Shoup SPHF, i.e., the client performs the following steps for each server S_j.

   (a) $k_{h0,j} \xleftarrow{R} KGen_H(L_{aux})$:

   $$k_{h0,j} = (\eta_{1,0,j}, \eta_{2,0,j}, \theta_{0,j}, \mu_{0,j}, \nu_{0,j}) \in_R \mathbb{Z}_p^{1\times 5}$$

   (b) $k_{p0,j} \leftarrow KGen_P(k_{h0,j}, L_{aux})$:

   $$\begin{aligned} k_{p0,j} &= \Gamma \odot k_{h0,j} \\ &= \begin{pmatrix} g_1 & 1 & g_2 & h & c \\ 1 & g_1 & 1 & 1 & d \end{pmatrix} \odot (\eta_{1,0,j}, \\ &\quad \eta_{2,0,j}, \theta_{0,j}, \mu_{0,j}, \nu_{0,j}) \\ &= \begin{pmatrix} g_1{}^{\eta_{1,0,j}} g_2{}^{\theta_{0,j}} h^{\mu_{0,j}} c^{\nu_{0,j}} \\ g_1{}^{\eta_{2,0,j}} d^{\nu_{0,j}} \end{pmatrix} \in \mathbb{G}^{2\times 1} \end{aligned}$$

   (c) $\ell_{0,j} = (P_0, S_j)$

   (d) $C_{0,j} \leftarrow Enc_{pk}^{CS}(\ell_{0,j}, pw; r_{0,j})$:

   $$r_{0,j} \xleftarrow{R} \mathbb{Z}_p$$

   $$\begin{aligned} u_{1,0,j} &= g_1{}^{r_{0,j}}, u_{2,0,j} = g_2{}^{r_{0,j}}, e_{0,j} = g_1{}^{pw} h^{r_{0,j}} \\ \xi_{0,j} &= H_k(\ell_{0,j}, u_{1,0,j}, u_{2,0,j}, e_{0,j}) \\ v_{0,j} &= (cd^{\xi_{0,j}})^{r_{0,j}} \\ C_{0,j} &= (\ell_{0,j}, u_{1,0,j}, u_{2,0,j}, e_{0,j}, v_{0,j}) \in \mathbb{G}^{1\times 5} \end{aligned}$$

   The client sends n messages to the servers, one $(C_{0,j}, k_{p0,j})$ for each server S_j.

2. Each server S_j runs the next step of the centralized Cramer-Shoup SPHF and generates a message (C_j, k_{pj}) sent to client P_0. Each server S_j also computes corresponding session key sk'_j. The client receives the following message from every server S_j.

   $$(C_j = (\ell_j, u_{1,j}, u_{2,j}, e_j, v_j) \in \mathbb{G}^{1\times 5},$$

   $$k_{pj} = \begin{pmatrix} g_1{}^{\eta_{1,j}} g_2{}^{\theta_j} h^{\mu_j} c^{\nu_j} \\ g_1{}^{\eta_{2,j}} d^{\nu_j} \end{pmatrix} \in \mathbb{G}^{2\times 1})$$

3. For each server S_j, client P_0 calculates k_{xj} and h_{0j} as follows and computes session key sk_j.

(a) $k_{xj} \leftarrow Hash_x(k_{h0,j}, L_{aux}, C_j)$:

$$\xi_j = H_k(\ell_j, u_{1,j}, u_{2,j}, e_j)$$
$$h_{xj} = \Theta^x_{aux}(C_j) \odot k_{h0,j}$$
$$= ((u_{1,j}), ((u_{1,j})^{\xi_j}), (u_{2,j}), (e_j/{g_1}^{pw}), (v_j)) \odot (\eta_{1,0,j}, \eta_{2,0,j}, \theta_{0,j}, \mu_{0,j}, \nu_{0,j})$$
$$= (u_{1,j})^{\eta_{1,0,j}} (u_{1,j})^{\xi_j \eta_{2,0,j}} (u_{2,j})^{\theta_{0,j}} (e_j/{g_1}^{pw})^{\mu_{0,j}} (v_j)^{\nu_{0,j}} \in \mathbb{G}$$

(b) $h_{0j} \leftarrow PHash_0(k_{pj}, L_{aux}, C_{0,j}, r_{0,j})$ with $\lambda^{0j} = (r_{0,j}, r_{0,j}\xi_{0,j})$:

$$h_{0j} = (\lambda^{0j} \odot k_{pj})$$
$$= (r_{0,j}, r_{0,j}\xi_{0,j}) \odot \begin{pmatrix} {g_1}^{\eta_{1,j}} {g_2}^{\theta_j} h^{\mu_j} c^{\nu_j} \\ {g_1}^{\eta_{2,j}} d^{\nu_j} \end{pmatrix}$$
$$= ({g_1}^{\eta_{1,j}} {g_2}^{\theta_j} h^{\mu_j} c^{\nu_j})^{r_{0,j}} ({g_1}^{\eta_{2,j}} d^{\nu_j})^{r_{0,j}\xi_{0,j}}$$
$$= (u_{1,0,j})^{\eta_{1,j}} (u_{2,0,j})^{\theta_j} (h)^{\mu_j r_{0,j}} (u_{1,0,j})^{\eta_{2,j}\xi_{0,j}} (v_{0,j})^{\nu_j} \in \mathbb{G}$$

(c) Client P_0 computes its session key sk_j for each server S_j:

$$sk_j = h_{xj} h_{0j}$$

3.4 The scheme of the servers' side

Public information: $\mathbb{G}$, p, g_1, g_2, g_3, g_4, g_5, g_6, h, c, d, H_k, L_{aux}, pk.

Input for servers S_j: T_{PW}, the output of the simulated Joint-Pedesen-RVSS for the password pw.

1. Client P_0 runs n times the first step of the centralized Cramer-Shoup SPHF and generates n messages, one for each server S_j. Each server S_j receives the following message $(C_{0,j}, k_{p0,j})$ from the client.

$$(C_{0,j} = (\ell_{0,j}, u_{1,0,j}, u_{2,0,j}, e_{0,j}, \nu_{0,j}) \in \mathbb{G}^{1\times 5},$$
$$k_{p0,j} = \begin{pmatrix} {g_1}^{\eta_{1,0,j}} {g_2}^{\theta_{0,j}} h^{\mu_{0,j}} c^{\nu_{0,j}} \\ {g_1}^{\eta_{2,0,j}} d^{\nu_{0,j}} \end{pmatrix} \in \mathbb{G}^{2\times 1})$$

2. Each server S_j performs the following steps and sends to client P_0 its output, i.e., a message $(C_j, k_{p,j})$ (Client P_0 receives n messages from the n servers).

(a) $k_{hj} \xleftarrow{R} KGen_H(L_{aux})$:

$$Joint - Pedersen - RVSS(g_1, g_2, g_3, g_4, g_5, g_6, h) \xrightarrow[t,n]{H_{1,j}, H_{2,j}, \Theta_j, M_j, N_j, R_j, \tilde{X}} [\eta_{1,j,i}, \eta_{2,j,i}, \theta_{j,i}, \mu_{j,i}, \nu_{j,i}, r_{j,i}, \tilde{x_{j,i}}, T_{H_{1,j}}, T_{H_{2,j}}, T_{\Theta_j}, T_{M_j}, T_{N_j}, T_{R_j}](V_{H_{1,j,k}}, V_{H_{2,j,k}}, V_{\Theta_{j,k}}, V_{M_{j,k}}, V_{N_{j,k}}, V_{R_{j,k}}) \{\eta_{1,j}, \eta_{2,j}, \theta_j, \mu_j, \nu_j, r_j\}$$
$$k_{hj} = (\eta_{1,j}, \eta_{2,j}, \theta_j, \mu_j, \nu_j) \in_R \mathbb{Z}_p^{1\times 5}$$

(b) $k_{pj} \leftarrow KGen_P(k_{hj}, L_{aux})$:

$$Shared - Exp[T_{H_{1,j}}, T_{\Theta_j}, T_{M_j}, T_{N_j}](g_1, g_2, h, c) \rightarrow (k_{pj,0} = {g_1}^{\eta_{1,j}} {g_2}^{\theta_j} h^{\mu_j} c^{\nu_j})$$
$$Shared - Exp[T_{H_{2,j}}, T_{N_j}](g_1, d) \rightarrow (k_{pj,1} = {g_1}^{\eta_{2,j}} d^{\nu_j})$$
$$k_{p_j} = \begin{pmatrix} k_{pj,0} \\ k_{pj,1} \end{pmatrix} = \begin{pmatrix} {g_1}^{\eta_{1,j}} {g_2}^{\theta_j} h^{\mu_j} c^{\nu_j} \\ {g_1}^{\eta_{2,j}} d^{\nu_j} \end{pmatrix} \in \mathbb{G}^{2\times 1}$$

(c) $\ell_j = (S_j, P_0)$

(d) $C_j \leftarrow Enc^{CS}_{pk}(\ell_j, pw; r_j)$:

$$\xi_j = H_k(\ell_j, u_{1,j}, u_{2,j}, e_j)$$
$$Shared - Exp[T_{R_j}](g_1) \rightarrow (u_{1,j} = {g_1}^{r_j})$$
$$Shared - Exp[T_{R_j}](g_2) \rightarrow (u_{2,j} = {g_2}^{r_j})$$
$$Shared - Exp[T_{PW}, T_{R_j}](g_1, h) \rightarrow (e_j = {g_1}^{pw} h^{r_j})$$
$$Shared - Exp[T_{R_j}](cd^{\xi_j}) \rightarrow (v_j = (cd^{\xi_j})^{r_j})$$
$$C_j = (\ell_j, u_{1,j}, u_{2,j}, e_j, v_j) \in \mathbb{G}^{1\times 5}$$

(e) Server S_j sends its message (C_j, k_{pj}) to Client P_0. The client computes corresponding session key sk_j.

3. To compute session key sk'_j, Server S_j calculates k'_{xj} and h'_{0j} as follows:

(a) $h'_{xj} \leftarrow PHash_x(k_{p0,j}, L_{aux}, C_j, r_j)$ with $\lambda^j = (r_j, r_j\xi_j)$:

$$Shared - Exp[T_{R_j}]({g_1}^{\eta_{1,0,j}} {g_2}^{\theta_{0,j}} h^{\mu_{0,j}} c^{\nu_{0,j}}, ({g_1}^{\eta_{2,0,j}} d^{\nu_{0,j}})^{\xi_j}) \rightarrow (h'_{xj} = ({g_1}^{\eta_{1,0,j}} {g_2}^{\theta_{0,j}} h^{\mu_{0,j}} c^{\nu_{0,j}})^{r_j} ({g_1}^{\eta_{2,0,j}} d^{\nu_{0,j}})^{\xi_j r_j})$$
$$h'_{xj} = ({g_1}^{\eta_{1,0,j}} {g_2}^{\theta_{0,j}} h^{\mu_{0,j}} c^{\nu_{0,j}})^{r_j} ({g_1}^{\eta_{2,0,j}} d^{\nu_{0,j}})^{\xi_j r_j})$$
$$= (u_{1,j})^{\eta_{1,0,j}} (u_{2,j})^{\theta_{0,j}} (h)^{\mu_{0,j} r_j} (u_{1,j})^{\eta_{2,0,j}\xi_j} (v_j)^{\nu_{0,j}} \in \mathbb{G}$$

(b) $h'_{0j} \leftarrow Hash_0(k_{hj}, L_{aux}, C_{0,j})$:

$$Shared - Exp[T_{H_{1,j}}, T_{H_{2,j}}, T_{\Theta_j}, T_{M_j}, T_{N_j}](u_{1,0,j}, {u_{1,0,j}}^{\xi_{0,j}}, u_{2,0,j}, e_{0,j}/{g_1}^{pw}, v_{0,j}) \rightarrow (h'_{0j} = (u_{1,0,j})^{\eta_{1,j}} (u_{1,0,j})^{\xi_{0,j}\eta_{2,j}} (u_{2,0,j})^{\theta_j} (e_{0,j}/{g_1}^{pw})^{\mu_j} (v_{0,j})^{\nu_j}) \in \mathbb{G}$$

(c) Server S_j locally computes its own session key:

$$sk'_j = h'_{xj} h'_{0j}$$

3.5 Correctness

The correctness of the scheme can be easily verified by checking if client P_0 and server S_j end with the same key, i.e., $sk_j = sk'_j$.

$$\begin{aligned}
sk_j &= h_{xj} h_{0j} \\
&= (u_{1,j})^{\eta_{1,0,j}} (u_{1,j})^{\xi_j \eta_{2,0,j}} (u_{2,j})^{\theta_{0,j}} (e_j / {g_1}^{pw})^{\mu_{0,j}} (v_j)^{\nu_{0,j}} \\
&\quad (u_{1,0,j})^{\eta_{1,j}} (u_{2,0,j})^{\theta_j} (h)^{\mu_j r_{0,j}} (u_{1,0,j})^{\eta_{2,j} \xi_{0,j}} (v_{0,j})^{\nu_j} \\
&= (u_{1,j})^{\eta_{1,0,j}} (u_{2,j})^{\theta_{0,j}} (e_j / {g_1}^{pw})^{\mu_{0,j}} (u_{1,j})^{\xi_j \eta_{2,0,j}} (v_j)^{\nu_{0,j}} \\
&\quad (u_{1,0,j})^{\eta_{1,j}} (u_{1,0,j})^{\eta_{2,j} \xi_{0,j}} (u_{2,0,j})^{\theta_j} (h)^{\mu_j r_{0,j}} (v_{0,j})^{\nu_j} \\
&= (u_{1,j})^{\eta_{1,0,j}} (u_{2,j})^{\theta_{0,j}} (h^{r_j})^{\mu_{0,j}} (u_{1,j})^{\xi_j \eta_{2,0,j}} (v_j)^{\nu_{0,j}} \\
&\quad (u_{1,0,j})^{\eta_{1,j}} (u_{1,0,j})^{\eta_{2,j} \xi_{0,j}} (u_{2,0,j})^{\theta_j} (h)^{\mu_j r_{0,j}} (v_{0,j})^{\nu_j}
\end{aligned}$$

$$\begin{aligned}
sk'_j &= h'_{xj} h'_{0j} \\
&= (u_{1,j})^{\eta_{1,0,j}} (u_{2,j})^{\theta_{0,j}} (h)^{\mu_{0,j} r_j} (u_{1,j})^{\eta_{2,0,j} \xi_j} (v_j)^{\nu_{0,j}} \\
&\quad (u_{1,0,j})^{\eta_{1,j}} (u_{1,0,j})^{\xi_{0,j} \eta_{2,j}} (u_{2,0,j})^{\theta_j} (e_{0,j} / {g_1}^{pw})^{\mu_j} (v_{0,j})^{\nu_j} \\
&= (u_{1,j})^{\eta_{1,0,j}} (u_{2,j})^{\theta_{0,j}} (h)^{\mu_{0,j} r_j} (u_{1,j})^{\eta_{2,0,j} \xi_j} (v_j)^{\nu_{0,j}} \\
&\quad (u_{1,0,j})^{\eta_{1,j}} (u_{1,0,j})^{\xi_{0,j} \eta_{2,j}} (u_{2,0,j})^{\theta_j} (h^{r_{0,j}})^{\mu_j} (v_{0,j})^{\nu_j} \\
&\therefore sk_j = sk'_j
\end{aligned}$$

4 Proof of security

Theorem 1. *If SPHF is secure and Cramer-Shoup encryption scheme is CCA-secure, then our scheme is a secure threshold PAKE scheme.*

Proof. In the distributed case we modify somewhat an adversary's power. First of all, the adversary does not have total control of the internal network of the servers. We assume that the servers have some authentication mechanism already in place in order to create secure channels between them. Moreover, we give the adversary the power to corrupt the servers. This will allow her to see their internal state and totally gain control of the server. We bound the number of servers that the adversary can corrupt by t .We assume the adversary to be *static*, i.e., the set of corrupted servers $S_1, \ldots S_t$ is decided in advance.

We prove security by showing a reduction to the security of the underlying centralized scheme, i.e., we show that if there is an adversary that is able to break our threshold PAKE, then we can build another adversary that breaks an instance of the centralized scheme. Let A be an adversary that tries to break the centralized version of the Cramer-Shoup SPHF^x. Adversary A invokes adversary A_{thresh}, which we assume is able to break the threshold PAKE version of the Cramer-Shoup SPHF. This execution is run in a "virtual world" in which all the exchanged messages are simulated.

Let us recall what it means to break a threshold PAKE protocol from [8]. Adversary A_{thresh} is allowed to invoke the commands *Execute*, *Send*, *Reveal* and *Test*. At the end for one of the executions in which she did not ask for *Reveal* query for the secret key, if adversary A_{thresh} asks a query *Test*, then the following happens. A coin b is flipped. If it lands $b = 0$, then the session key sk is returned to adversary A_{thresh}, If it lands $b = 1$, then a random session key is returned. Adversary A_{thresh} wins if she recognizes the right key with probability substantially better than 1/2.

The centralized adversary A behaves as a simulator which creates a virtual distribute world for A_{thresh}. The centralized adversary A installs a fake password. For each execution of the distributed scheme, the centralized adversary A requests n sessions of the Cramer-Shoup SPHF protocol she is trying to break. She uses these n sessions to "embed" n sessions in the distributed protocol, one for each server.

(1) Execute command: When A_{thresh} invokes an *Execute* command, Adversary A invokes n *Execute* in the centralized world. This gives n transcripts ($msg_{j_1} = (C_{0,j}, k_{p0,j}), msg_{j_2} = (C_j, k_{pj})$). The simulator have to create a view in which the ith component match msg_{j_1}, msg_{j_2}. For msg_{j_1}, this is not a problem since the simulator controls the client. But for msg_{j_2} we need to show that the transcript of the distributed scheme can be manipulated to hit this specific value. Let us see the details (with the steps enumeration related to Section 3.4), adversary A:

1: simulates the first n messages of the client using the messages $msg_{j,1}$

2a: executes in the virtual distributed world the instances of Joint-Pedersen-RVSS to choose at random $\eta_{1,j}$, $\eta_{2,j}$, θ_j, μ_j, ν_j.

2b: simulates the executions of Shared-Exp in order to "hit" the right message k_{pj} from the transcript $msg_{j,2}$

2c: follows the real scheme.

2d: simulates the executions of Shared-Exp in order to "hit" the right message C_j from the transcript $msg_{j,2}$

2e: follows the real scheme.

3a: executes normal Shared-Exp for value h'_{xj}

3b: executes normal Shared-Exp for value h'_{0j}

3c: computes the faked secret key $\overline{sk'_j}$

(2) Send command: Let us go into the details for each type of message that adversary A_{thresh} can send:

1st msg ($P_0 \to S_j$): If A_{thresh} sends a message ($C_{0,j}$, $k_{p0,j}$) to server S_j, then A forwards the request to the real centralized server, and gets a reply (C_j, k_{pj}). This reply will be embedded in the virtual distributed world "hitting" these values as seen in the previous steps.

2st msg ($S_j \to P_0$): If A_{thresh} sends a message (C_j, k_{pj}) to the client, then A does not reply anything, but she forwards the message to the real centralized server.

(3) Reveal command: If adversary A_{thresh} asks a *Reveal* query on any server, adversary A will ask a *Reveal* query on the corresponding centralized session and forward the answer to A_{thresh}.

(4) Test command: When adversary A asks her $Test$ query in the centralized case and gets a string which is passed on to A_{thresh}. Then A answers whatever A_{thresh} answers (i.e., true key or random key).

The advantage of A is exactly the same as the advantage of A_{thresh}. Indeed the simulated view of A_{thresh} is exactly the same as the view of a real execution of the scheme (this is because the simulation of Shared-Exp is perfect [8]). □

References

[1] S. Bellovin, M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks. *IEEE Symposium on Security and Privacy*, 1992.

[2] R. Cramer and V.Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Eurocrypt 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45-64. Springer-Verlag, 2002.

[3] W. Ford, B. Kaliski. Server-assisted generation of a strong secret from a password. *Proceedings of the 5th IEEE International Workshop on Enterprise Security*, 2000.

[4] R. Gennaro, Y. Lindell. A Framework for Password-Based Authenticated Key Exchange. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524-543. Springer-Verlag, 2003.

[5] F. Kiefer, M. Manulis. Distributed Smooth Projective Hashing and Its Application to Two-Server Password Authenticated Key Exchange. In *ACNS 2014*, volume 8479 of *Lecture Notes in Computer Science*, pages 199-216. Springer-Verlag, 2014.

[6] J. Katz, R. Ostrovsky, M. Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In *EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475-494. Springer-Verlag, 2001.

[7] P. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange. In *Eurocrypt 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 385-400. Springer-Verlag, 2002.

[8] M. D. Raimondo, R. Gennaro. Provably secure threshold password-authenticated key exchange. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 507-523. Springer-Verlag, 2003.

[9] Fabrice Ben Hamouda, Olivier Blazy, Celine Chevalier, David Pointcheval, and Damien Vergnaud. Efficient uc-secure authenticated key-exchange for algebraic languages.In Kaoru Kurosawa and Goichiro Hanaoka, editors, PKC 2013: 16th International Conference on Practice and Theory in Public-Key Cryptography, volume 7778 of Lecture Notes in Computer Science, pages 272 291. Springer, 2013.

[10] F.Benhamouda,O.Blazy,C.Chevalier,D.Pointcheval, andD.Vergnaud.NewSmoothProjectiveHashFunctions and One-Round Authenticated Key Exchange. Cryptology ePrint Archive, Report 2013/034, 2013. http:// eprint.iacr.org/. 1, 2, 3, 4, 5, 15

[11] T. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing. In CRYPTO '91, Springer LNCS 576,pp.129-140,1991.

[12] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, The (in)security of distributed key generation in dlog-based cryptosystems, in: Advances in Cryptology-Proceedings of EUROCRYPT '99, in: Lecture Notes in Comput. Sci., vol. 1592, Springer-Verlag, 1999, pp.295-310.