

# A Secure Threshold Anonymous Password-Authenticated Key Exchange Protocol

SeongHan Shin<sup>1</sup>, Kazukuni Kobara<sup>1</sup>, and Hideki Imai<sup>1,2</sup>

<sup>1</sup> Research Center for Information Security (RCIS),  
National Institute of Advanced Industrial Science and Technology (AIST),  
1-18-13, Sotokannda, Chiyoda-ku, Tokyo 101-0021 Japan  
{seonghan.shin, k-kobara, h-imai}@aist.go.jp

<sup>2</sup> Chuo University,  
1-13-27, Haruhi, Bunkyo-ku, Tokyo 112-8551 Japan

**Abstract.** At Indocrypt 2005, Viet et al., [20] have proposed an anonymous password-authenticated key exchange (PAKE) protocol and its threshold construction both of which are designed for client's password-based authentication and anonymity against a passive server, who does not deviate the protocol. In this paper, we first point out that their threshold construction is completely insecure against off-line dictionary attacks. For the threshold  $t > 1$ , we propose a secure threshold anonymous PAKE (for short, TAP) protocol with the number of clients  $n$  upper-bounded, such that  $n \leq 2\sqrt{N-1} - 1$ , where  $N$  is a dictionary size of passwords. We also show that the TAP protocol provides semantic security of session keys in the random oracle model, with the reduction to the computational Diffie-Hellman problem, as well as anonymity against a passive server. For the threshold  $t = 1$ , we propose an efficient anonymous PAKE protocol that significantly improves efficiency in terms of computation costs and communication bandwidth compared to the original (not threshold) anonymous PAKE protocol [20].

## 1 Introduction

In 1976, Diffie and Hellman published their seminal paper that introduced how to share a secret over public networks [9]. Since then, many researchers have tried to design secure cryptographic protocols for realizing secure channels. These protocols are necessary because application-oriented protocols are frequently developed assuming the existence of such secure channels. In the 2-party setting (e.g., a client and a server), this can be achieved by an authenticated key exchange (AKE) protocol at the end of which the two parties authenticate each other and share a common and temporal session key to be used for subsequent cryptographic algorithms (e.g., AES-CBC or MAC). For authentication, the parties typically share some information in advance. The shared information may be the form of high-entropy cryptographic keys: either a secret key that can be used for symmetric-key encryption or message authentication

code (e.g., [7,16]), or public keys (while the corresponding private keys are kept secret) which can be used for public-key encryption or digital signatures (e.g., [10,22,2,16,12]).

In practice, low-entropy human-memorable passwords such as 4-digit pin-code or alphanumerical passwords are commonly used rather than high-entropy keys because of its convenience in use. Many password-based AKE protocols have been extensively investigated for a long time where a client remembers a short password and the corresponding server holds the password or its verification data that is used to verify the client's knowledge of the password. However, one should be careful about two major attacks on passwords: on-line and off-line dictionary attacks. The on-line dictionary attack is a series of exhaustive searches for a secret performed on-line, so that an adversary can sieve out possible secret candidates one by one communicating with the target party. In contrast, the off-line dictionary attack is performed off-line in parallel where an adversary exhaustively enumerates all possible secret candidates, in an attempt to determine the correct one, by simply guessing a secret and verifying the guessed secret with recorded transcripts of a protocol. While on-line attacks are applicable to all of the password-based protocols equally, they can be prevented by letting a server take appropriate intervals between invalid trials. But, we cannot avoid off-line attacks by such policies, mainly because the attacks can be performed off-line and independently of the party.

### 1.1 Password-Authenticated Key Exchange and Anonymity

In 1992, Bellare and Merritt [4] discussed an interesting problem about how to design a secure password-only protocol where a client remembers his/her password only (without any device and any additional assumption) and the counterpart server has password verification data. Their proposed protocols are good examples (though some are turned out insecure) that a combination of symmetric and asymmetric cryptographic techniques can prevent an adversary from verifying a guessed password (i.e., doing off-line dictionary attacks). Later, their AKE protocols have formed the basis for what we call Password-Authenticated Key Exchange (PAKE) protocols. Such protocols have been in standardization of IEEE P1363.2 [11].

In PAKE protocols, a client should send his/her identity clearly in order to authenticate each other and share a master-secret that may be the Diffie-Hellman key or a shared secret to be used for generating authenticators and session keys. Let us suppose an adversary who fully controls the networks. Though the adversary cannot impersonate any party in PAKE protocols with non-negligible probability, it is easy to collect a client's personal information about the communication history itself (e.g., history of access to ftp servers, web-mail servers, Internet banking servers or shopping mall servers). These information may reflect the client's life pattern and sometimes can be used for spam mails. For this problem, Viet et al., [20] have proposed an anonymous PAKE protocol and its

threshold construction<sup>1</sup> that simply combine a PAKE protocol [1] for generating secure channels with an Oblivious Transfer (OT) protocol [19,8] for client's anonymity. The anonymity is guaranteed against an outside adversary as well as a passive server, who follows the protocol honestly but it is curious about identity of client involved with the protocol. They also gave an application for a company's public bulletin board to which any employee can upload opinions in a password-authenticated and anonymous way. As discussed in [20], their (not threshold) anonymous PAKE protocol can not provide anonymity against an active server, who deviates the protocol by changing messages at its own (see Section 5 of [20]). Though they did not mention anything about their threshold construction, it may prevent an active server from obtaining information on the client's identity since any client can blend him/herself to the subgroup.

## 1.2 Our Contributions

Partly motivated from Nguyen's insights [14] on the relationship between PAKE protocols and other cryptographic primitives, we carefully revisit Viet et al's anonymous PAKE protocols [20]. In this paper, we first point out that Viet et al's threshold anonymous PAKE protocol is insecure against off-line dictionary attacks. For the threshold  $t > 1$ , we propose a secure threshold anonymous PAKE (for short, TAP) protocol that provides not only semantic security of session keys in the random oracle model with the reduction to the computational Diffie-Hellman problem but also anonymity against a passive server, who does not deviate the protocol but is curious about the clients' identities. We also give the condition on the number of clients  $n$ , such that  $n \leq 2\sqrt{N-1} - 1$ , for the optimal security result against on-line dictionary attacks where  $N$  is a dictionary size of passwords. For the threshold  $t = 1$ , we propose an efficient anonymous PAKE protocol that can be easily obtained from the TAP protocol. The resultant protocol significantly improves efficiency in terms of computation costs and communication bandwidth compared to the original (not threshold) anonymous PAKE protocol [20].

## 1.3 Organization

This paper is organized as follows. In the next section, we show that the previous threshold anonymous PAKE protocol is insecure against off-line attacks. Section 3 is assigned to security model. In Section 4, we propose a secure threshold anonymous PAKE (TAP) protocol. Section 5 is devoted to its security proofs, followed by discussion about the condition on  $n$  in Section 6. For the threshold  $t = 1$ , we also propose an efficient anonymous PAKE protocol in Section 7. Finally, we conclude in Section 8.

<sup>1</sup> Here, the "threshold" number of clients are involved with the protocol. In a different context, MacKenzie et al., [13] proposed a threshold PAKE protocol where the "threshold" number of servers collaborates one another to resist against compromise of the password verification data. However, the collaborations in the former/latter protocols assume secure channels among clients/servers, respectively.

## 2 The Previous Threshold Anonymous PAKE Protocol

In this section, we first give some notation to be used throughout this paper. Then we explain how the previous threshold anonymous PAKE protocol [20,21] works and show its insecurity against off-line dictionary attacks.

### 2.1 Notation

Let  $\mathbb{G}_p$  be a finite, cyclic group of prime order  $p$  and  $g$  be a generator of  $\mathbb{G}_p$ , whose elements are quadratic residues modulo  $p$ . Let  $h$  be another generator of  $\mathbb{G}_p$  so that its discrete logarithm problem with  $g$  (i.e., computing  $b = \log_g h$ ) should be hard. The parameter  $(\mathbb{G}_p, p, g, h)$  is given as public information. In the aftermath, all the subsequent arithmetic operations are performed in modulo  $p$  unless otherwise stated.

Let  $l$  denote the security parameter for hash functions. Let  $N$  be a dictionary size of passwords. Let  $\{0, 1\}^*$  denote the set of finite binary strings and  $\{0, 1\}^l$  the set of binary strings of length  $l$ . If  $D$  is a set, then  $d \stackrel{R}{\leftarrow} D$  indicates the process of selecting  $d$  at random and uniformly over  $D$ . Let "||" denote the concatenation of bit strings in  $\{0, 1\}^*$ . Let " $\oplus$ " denote the exclusive-OR (XOR) operation of bit strings. The hash functions  $\mathcal{F}$  and  $\mathcal{F}'$  are full-domain hash (FDH) functions, mapping  $\{0, 1\}^*$  to  $\mathbb{Z}_p^*$ . While  $\mathcal{G} : \{0, 1\}^* \rightarrow \mathbb{G}_p$  is another FDH function, the others are denoted  $\mathcal{H}_k : \{0, 1\}^* \rightarrow \{0, 1\}^l$ , for  $k = 1, 2$  and  $3$ , where  $\mathcal{G}$  and  $\mathcal{H}_k$  are distinct secure one-way hash functions. Let  $C$  and  $S$  be the identities of a set of all clients and server, respectively, with each  $ID \in \{0, 1\}^*$ .

### 2.2 Protocol Description

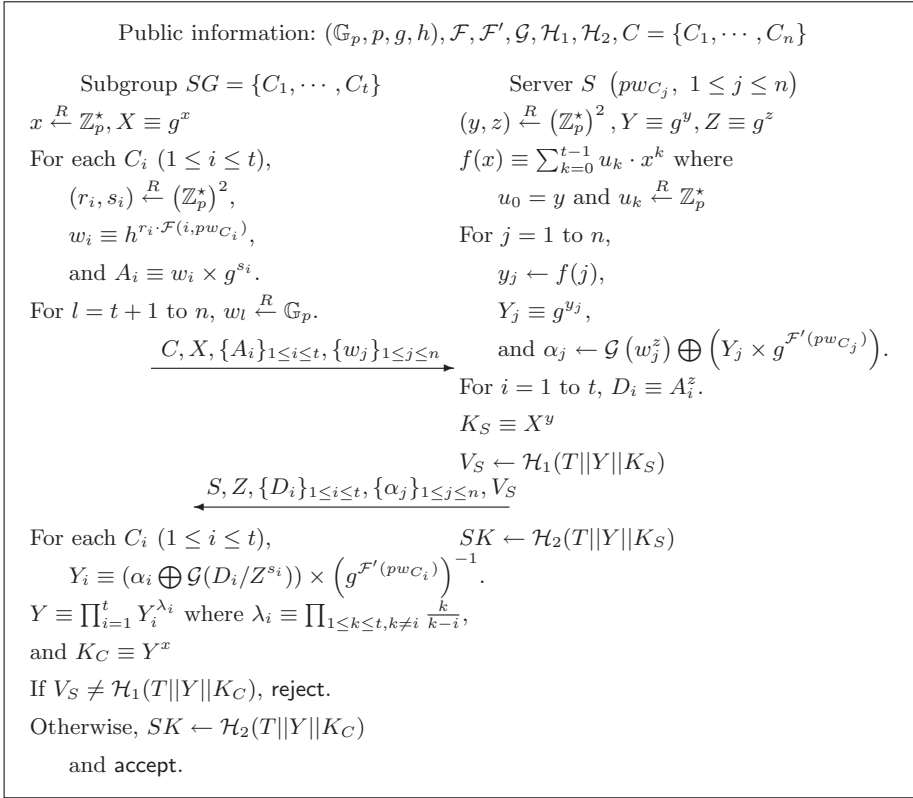
Here we describe the threshold anonymous PAKE (TA-PAKE) protocol [20,21] where any subgroup  $SG$ , consisting of at least  $t$  ( $t \leq n$ ) clients among  $n$  clients, generates a session key with server  $S$  in a password-authenticated and anonymous way.<sup>2</sup> We assume that each client in the subgroup are connected via secure channels. See Fig. 1 for graphical description of the TA-PAKE protocol.

#### Step 1

- 1.1 By collaborating with one another, the subgroup  $SG$  chooses a random number  $x$  from  $\mathbb{Z}_p^*$  and computes  $X \equiv g^x$ .
- 1.2 Each client  $C_i$  ( $1 \leq i \leq t$ ) chooses two random numbers  $(r_i, s_i) \stackrel{R}{\leftarrow} (\mathbb{Z}_p^*)^2$ , and then computes  $w_i \equiv h^{r_i \cdot \mathcal{F}(i, pw_{C_i})}$  and  $A_i \equiv w_i \times g^{s_i}$  where  $i$  and  $pw_{C_i}$  are the index and the password, respectively, for client  $C_i$ . The  $r_i$  and  $s_i$  are kept secret by  $C_i$ .

---

<sup>2</sup> The only difference of [21] from [20] is that the subgroup  $SG$  chooses  $w_l \stackrel{R}{\leftarrow} \mathbb{G}_p$ , for  $t + 1 \leq l \leq n$ , and sends  $\{w_j\}_{1 \leq j \leq n}$  along with other values in the first flow. In fact, the TA-PAKE protocol of [20] doesn't work correctly since server  $S$  has no idea on  $w_j$ .



**Fig. 1.** The threshold anonymous PAKE (TA-PAKE) protocol [20,21] where  $T = C||S||X||Z||\{D_i\}_{1 \leq i \leq t}||\{\alpha_j\}_{1 \leq j \leq n}$

**1.3** The subgroup  $SG$  chooses  $n - t$  random numbers  $w_l$  from  $\mathbb{G}_p$ , and then sends  $C, X, \{A_i\}_{1 \leq i \leq t}$  and  $\{w_j\}_{1 \leq j \leq n}$  to server  $S$ .

**Step 2**

**2.1** The server chooses two random numbers  $(y, z) \xleftarrow{R} (\mathbb{Z}_p^*)^2$  and computes  $(Y \equiv g^y, Z \equiv g^z)$  where the exponent  $y$  is distributed as shares by using Shamir's secret sharing scheme [15]. Specifically, server  $S$  generates the respective share  $f(j)$ , for  $n$  clients, from a random polynomial  $f(x)$  of degree  $t - 1$  with coefficients  $u_k$  ( $1 \leq k \leq t - 1$ ) in  $\mathbb{Z}_p^*$

$$f(x) \equiv \sum_{k=0}^{t-1} u_k \cdot x^k \tag{1}$$

and sets  $u_0 = y$ .

**2.2** For  $j$  ( $1 \leq j \leq n$ ), server  $S$  computes  $Y_j \equiv g^{f(j)}$  and  $\alpha_j \leftarrow \mathcal{G}(w_j^z) \oplus (Y_j \times g^{\mathcal{F}'(pw_{C_j})})$ .

- 2.3 For  $i$  ( $1 \leq i \leq t$ ), server  $S$  computes  $D_i \equiv A_i^z$ .
- 2.4 The server computes  $K_S \equiv X^y$ , from which its authenticator  $V_S$  and session key  $SK$  are derived as follows:  $V_S \leftarrow \mathcal{H}_1(C||S||X||Z||\{D_i\}_{1 \leq i \leq t}||\{\alpha_j\}_{1 \leq j \leq n}||Y||K_S)$  and  $SK \leftarrow \mathcal{H}_2(C||S||X||Z||\{D_i\}_{1 \leq i \leq t}||\{\alpha_j\}_{1 \leq j \leq n}||Y||K_S)$ . Then, server  $S$  sends  $S, Z, \{D_i\}_{1 \leq i \leq t}, \{\alpha_j\}_{1 \leq j \leq n}$  and  $V_S$  to subgroup  $SG$ .

**Step 3**

- 3.1 Each client  $C_i$  ( $1 \leq i \leq t$ ) extracts  $Y_i \equiv (\alpha_i \oplus \mathcal{G}(D_i/Z^{s_i})) \times (g^{\mathcal{F}'(pw_{C_i})})^{-1}$ .
- 3.2 By collaborating with one another, subgroup  $SG$  recovers  $Y$  from  $Y_i$  and computes  $K_C \equiv Y^x$ . Note that the  $Y$  can be reconstructed from the shares of any qualified subgroup of clients by Lagrange interpolation.
- 3.3 If  $V_S$  is valid, subgroup  $SG$  computes a session key  $SK$  as follows:  $SK \leftarrow \mathcal{H}_2(C||S||X||Z||\{D_i\}_{1 \leq i \leq t}||\{\alpha_j\}_{1 \leq j \leq n}||Y||K_C)$ . Otherwise, it terminates.

**2.3 Insecurity of TA-PAKE Protocol**

We show that the TA-PAKE protocol [20,21] is insecure against off-line dictionary attacks. First we suppose that an adversary  $\mathcal{A}$  impersonates the subgroup  $SG$  without knowing any password.

**Step 1'**

- 1.1 An adversary  $\mathcal{A}$  chooses a random number  $x$  from  $\mathbb{Z}_p^*$  and computes  $X \equiv g^x$ , and also chooses  $n$  random numbers  $w_j \xleftarrow{R} \mathbb{G}_p$ , for  $1 \leq j \leq n$ .
- 1.2 For each client  $C_i$  ( $1 \leq i \leq t$ ), adversary  $\mathcal{A}$  chooses a random number  $s_i \xleftarrow{R} \mathbb{Z}_q^*$  and then computes  $A_i \equiv w_i \times g^{s_i}$ . The adversary sends  $C, X, \{A_i\}_{1 \leq i \leq t}$  and  $\{w_j\}_{1 \leq j \leq n}$  to server  $S$ .

**Step 3'**

- 3.1 After receiving the message from server  $S$ , adversary  $\mathcal{A}$  performs the following: compute  $Y'_i$ , as the honest client  $C_i$  of subgroup  $SG$  would do, with all of the possible password candidates  $pw'_{C_i}$  and store  $N$  different  $Y'_i$ , for each client  $C_i$  ( $1 \leq i \leq t$ ).

$$Y'_i \equiv \left( \alpha_i \oplus \mathcal{G}(D_i/Z^{s_i}) \right) \times \left( g^{\mathcal{F}'(pw'_{C_i})} \right)^{-1} \tag{2}$$

- 3.2 With  $tN$  different  $Y'_i$ , the adversary recovers  $Y' \equiv \prod_{i=1}^t Y_i'^{\lambda_i}$  and the latter is used to compute  $K'_C \equiv Y'^x$ . Finally, adversary  $\mathcal{A}$  can find out the correct  $\{pw'_{C_1}, \dots, pw'_{C_t}\}$  by checking whether a subgroup of password candidates satisfies  $V_S = \mathcal{H}_1(C||S||X||Z||\{D_i\}_{1 \leq i \leq t}||\{\alpha_j\}_{1 \leq j \leq n}||Y'||K'_C)$  or not. Note that each subgroup guarantees a unique polynomial  $f'(x)$  of degree  $t - 1$ .

In the worst case, adversary  $\mathcal{A}$  can find out  $\{pw'_{C_1}, \dots, pw'_{C_t}\}$  after  $N^t$  trials. Though the number of trials goes exponentially with the threshold  $t$ , one can see that if  $t$  is small it is easy for an adversary to get the correct passwords.

More importantly, the above attack implies that a legitimate client in  $C$  can also obtain all passwords of the other clients with the linear trials. Suppose that there are two legitimate clients  $C_1$  and  $C_3$  who make up a subgroup  $SG = \{C_1, C_2, C_3\}$ . After running the TA-PAKE protocol, as an adversary would do in the above, with server  $S$ ,  $C_1$  and  $C_3$  can know the password of  $C_2$  by checking possible  $N$  password candidates in the same way as above. By repeating this off-line dictionary attack  $n - 2$  times,  $C_1$  and  $C_3$  find out all passwords of the remaining clients in  $C$ .

### 3 The Model and Security Notions

In this section, we introduce the model based on [5,3], security notions and the underlying mathematical assumption.

**The Model.** We consider  $SG$  (i.e., a subgroup of  $C$ ) and  $S$  as two parties that participate in the key exchange protocol  $P$ . Each of  $SG$  and  $S$  may have several instances called oracles involved in distinct, possibly concurrent, executions of  $P$ . We denote  $SG$  (resp.,  $S$ ) instances by  $SG^\mu$  (resp.,  $S^\nu$ ) where  $\mu, \nu \in \mathbb{N}$ , or by  $U$  in case of any instance. In the TAP protocol, each client  $C_i$  ( $1 \leq i \leq n$ ) of  $C$  and server  $S$  share a low-entropy secret  $pw_{C_i}$  drawn from a small dictionary of password  $\mathbb{D}_{\text{Password}}$ , whose cardinality is  $N$ . Here we assume that an adversary  $\mathcal{A}$  is not any client and server (i.e.,  $\mathcal{A} \notin \{C, S\}$ ). However, the adversary has the entire control of the network during the protocol execution which can be represented by allowing  $\mathcal{A}$  to ask several queries to oracles. Let us show the capability of adversary  $\mathcal{A}$  each query captures:

- **Execute**( $SG^\mu, S^\nu$ ): This query models passive attacks, where the adversary gets access to honest executions of  $P$  between the instances  $SG^\mu$  and  $S^\nu$  by eavesdropping.
- **Send**( $U, m$ ): This query models active attacks by having  $\mathcal{A}$  send a message to instance  $U$ . The adversary  $\mathcal{A}$  gets back the response  $U$  generates in processing the message  $m$  according to the protocol  $P$ . A query **Send**( $SG^\mu, \text{Start}$ ) initializes the key exchange protocol, and thus the adversary receives the first flow.
- **Reveal**( $U$ ): This query handles the misuse of the session key (e.g., use in a weak symmetric-key encryption) by any instance  $U$ . The query is only available to  $\mathcal{A}$ , if the instance actually holds a session key, and at that case the key is released to  $\mathcal{A}$ .
- **Test**( $U$ ): This oracle is used to see whether or not the adversary can obtain some information on the session key by giving a hint on the key. The **Test**-query can be asked at most once by the adversary  $\mathcal{A}$  and is only available to  $\mathcal{A}$  if the instance  $U$  is "fresh" in that the session key is not obviously known to the adversary. This query is answered as follows: one flips a private coin

$b \in \{0, 1\}$  and forwards the corresponding session key  $SK$  ( $\text{Reveal}(U)$  would output) if  $b = 1$ , or a random value with the same size except the session key if  $b = 0$ .

**Security Notions.** The adversary  $\mathcal{A}$  is provided with random coin tosses, some oracles and then is allowed to invoke any number of queries as described above, in any order. The aim of the adversary is to break the privacy of the session key (a.k.a., semantic security) or the authentication of the parties in the context of executing  $P$ .

The AKE security is defined by the game  $\text{Game}^{\text{ake}}(\mathcal{A}, P)$ , in which the ultimate goal of the adversary is to guess the bit  $b$  involved in the  $\text{Test}$ -query by outputting this guess  $b'$ . We denote the AKE advantage, by  $\text{Adv}_P^{\text{ake}}(\mathcal{A}) = 2 \Pr[b = b'] - 1$ , as the probability that  $\mathcal{A}$  can correctly guess the value of  $b$ . The protocol  $P$  is said to be  $(t, \varepsilon)$ -AKE-secure if  $\mathcal{A}$ 's advantage is smaller than  $\varepsilon$  for any adversary  $\mathcal{A}$  running time  $t$ .

Another goal is to consider unilateral authentication of either  $SG$  ( $SG\text{-auth}$ ) or  $S$  ( $S\text{-auth}$ ) wherein the adversary impersonates a party. We denote by  $\text{Succ}_P^{\text{SG-auth}}(\mathcal{A})$  (resp.,  $\text{Succ}_P^{\text{S-auth}}(\mathcal{A})$ ) the probability that  $\mathcal{A}$  successfully impersonates an  $SG$  instance (resp., an  $S$  instance) in an execution of  $P$ , which means that  $S$  (resp.,  $SG$ ) agrees on a key while the latter is shared with no instance of  $SG$  (resp.,  $S$ ). A protocol  $P$  is said to be  $(t, \varepsilon)$ -Auth-secure if  $\mathcal{A}$ 's success probability for breaking either  $SG\text{-auth}$  or  $S\text{-auth}$  is smaller than  $\varepsilon$  for any adversary  $\mathcal{A}$  running time  $t$ .

By following the definition of anonymity from [20], we can say that a protocol  $P$  is *anonymous* if a passive server cannot get any information about clients' identities (in  $SG$ ) involved with the protocol, whereas the subgroup  $SG$  establishes a session key with the server. In other words, any subgroup can prove that it consists of legitimate members of the set  $C$  by sending its authenticator at the end of the protocol. Nevertheless, the server does not know who they are.

### 3.1 Computational Diffie-Hellman Assumption

A  $(t_1, \varepsilon_1)$ -CDH $_{g, \mathbb{G}_p}$  attacker, in a finite cyclic group  $\mathbb{G}_p$  of prime order  $p$  with  $g$  as a generator, is a probabilistic machine  $\mathcal{B}$  running in time  $t_1$  such that its success probability  $\text{Succ}_{g, \mathbb{G}_p}^{\text{cdh}}(\mathcal{B})$ , given random elements  $g^x$  and  $g^y$  to output  $g^{xy}$ , is greater than  $\varepsilon_1$ . We denote by  $\text{Succ}_{g, \mathbb{G}_p}^{\text{cdh}}(t_1)$  the maximal success probability over every adversaries running within time  $t_1$ . The CDH-Assumption states that  $\text{Succ}_{g, \mathbb{G}_p}^{\text{cdh}}(t_1) \leq \varepsilon_1$  for any  $t_1/\varepsilon_1$  not too large.

## 4 A Secure Threshold Anonymous PAKE (TAP) Protocol

In this section, we propose a secure threshold anonymous PAKE (for short, TAP) protocol that has the following properties: 1) semantic security of session keys against an outside adversary; and 2) anonymity against a passive server, who follows the protocol honestly but is curious about clients' identities involved with



the protocol. Here we assume that all clients  $C_i$  ( $1 \leq i \leq n$ ) of the set  $C$  has registered their passwords  $pw_{C_i}$  to a server  $S$  and the server holds the password verification data in an asymmetric form (i.e.,  $h^{\mathcal{F}(i,pw_{C_i})}$ ). For simplicity, we assign the clients consecutive integer  $i$  ( $1 \leq i \leq n$ ) where  $C_i$  can be regarded as the  $i$ -th client of  $C$ . In the TAP protocol, any subgroup  $SG$  consisting of at least  $t$  ( $t > 1$ ) clients wants to share a session key securely and anonymously with server  $S$  (see Fig. 2).

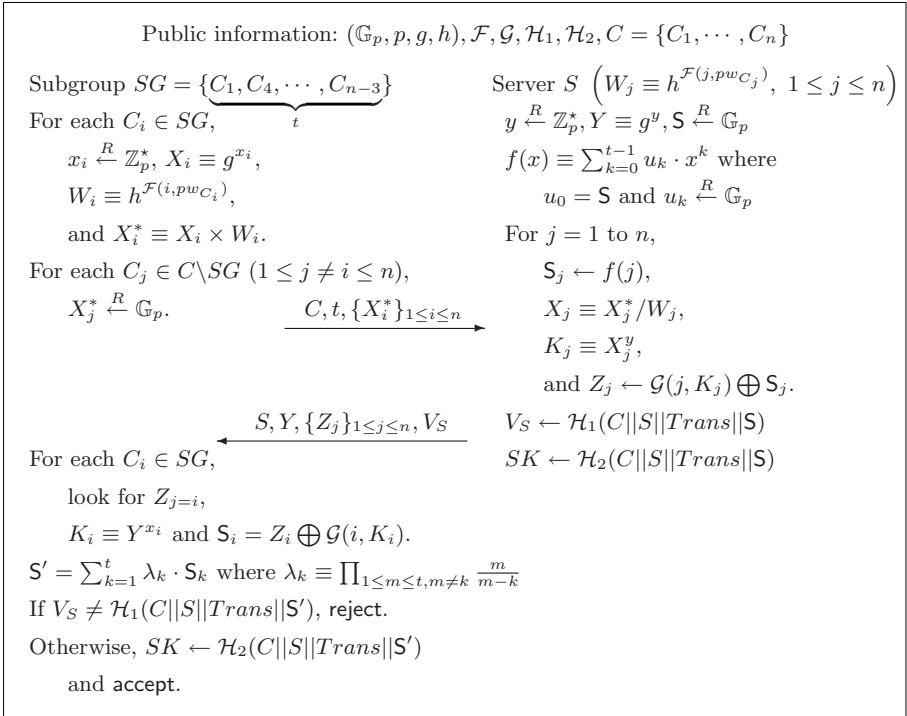
**RATIONALE.** A naive approach for secure threshold anonymous PAKE protocol is performing the existing (not threshold) anonymous PAKE protocol up to  $t$  times. This apparently entails a lot of messages to be exchanged between subgroup  $SG$  and server  $S$ . In order to construct efficiently, the TAP protocol has the following rationale. The first is that, instead of client’s password itself, the output of  $\mathcal{F}(i,pw_{C_i})$  is used as an exponent in order to compute the verification data  $W_i$  as in [20]. In fact, this plays a very important role when  $t = 1$  (see Section 7) in that an adversary is enforced to make an on-line dictionary attack on a specific client, not the others. The second is that server generates only one Diffie-Hellman public value and its exponent is used to compute all of the possible Diffie-Hellman key  $K_j$ . As we will show in the proof, this is the reason why an adversary can get a factor  $n$  in the second term of the security result of Theorem 1. The third is that server sends  $\{Z_j\}_{1 \leq j \leq n}$  by encrypting a share of the secret  $S$  with the hash of each Diffie-Hellman key. This is enough to guarantee clients’ anonymity against an honest-but-curious server (see Theorem 2 in Section 5).

**Step 1**

- 1.1 Each client  $C_i$ , who belongs to the subgroup  $SG$ , chooses a random number  $x_i$  from  $\mathbb{Z}_p^*$  and computes the Diffie-Hellman public value  $X_i \equiv g^{x_i}$ . The client  $C_i$  also computes the password verification data  $W_i \equiv h^{\mathcal{F}(i,pw_{C_i})}$  where  $i$  and  $pw_{C_i}$  are the index and the password, respectively, for  $C_i$ . The  $W_i$  is used to mask  $X_i$ , so that its resultant value  $X_i^*$  can be obtained in a way of  $X_i^* \equiv X_i \times W_i$ . The chosen  $x_i$  is kept secret by  $C_i$ .
- 1.2 By collaborating with one another, subgroup  $SG$  (or any client  $C_i$ ) chooses  $X_j^* \stackrel{R}{\leftarrow} \mathbb{G}_p$  for each  $C_j$  ( $1 \leq j \neq i \leq n$ ), who belongs to  $C$  but not to  $SG$ . Then the subgroup sends the threshold  $t$ , the masked public values  $\{X_i^*\}_{1 \leq i \leq n}$ , to the server, together with the set  $C$  of clients’ identities.

**Step 2**

- 2.1 The server  $S$  chooses a random number  $y$  from  $\mathbb{Z}_p^*$  and a random secret  $S$  from  $\mathbb{G}_p$ , and computes its Diffie-Hellman public value  $Y \equiv g^y$ . The secret  $S$  is distributed as shares by using Shamir’s  $(t, n)$  secret sharing scheme [15]. Specifically, server  $S$  generates the respective share  $f(j)$ , for all clients, from a polynomial  $f(x) \equiv \sum_{k=0}^{t-1} u_k \cdot x^k$  with  $u_0 = S$  and coefficients  $u_k$  ( $1 \leq k \leq t - 1$ ) randomly chosen from  $\mathbb{G}_p$ .
- 2.2 For the received  $X_j^*$  ( $1 \leq j \leq n$ ), server  $S$  computes  $X_j \equiv X_j^*/W_j$  and the Diffie-Hellman key  $K_j \equiv X_j^y$ . The  $Z_j$  is derived from XORing  $S_j$



**Fig. 2.** A secure threshold anonymous PAKE (TAP) protocol where the threshold  $t > 1$  and  $Trans = t || \{X_i^*\}_{1 \leq i \leq n} || Y || \{Z_j\}_{1 \leq j \leq n}$

and the hashed output of index  $j$  and  $K_j$ :  $Z_j \leftarrow \mathcal{G}(j, K_j) \oplus S_j$  where  $S_j \leftarrow f(j)$ .

**2.3** Also server  $S$  generates an authenticator  $V_S \leftarrow \mathcal{H}_1(C || S || t || \{X_i^*\}_{1 \leq i \leq n} || Y || \{Z_j\}_{1 \leq j \leq n} || S)$  and a session key  $SK \leftarrow \mathcal{H}_2(C || S || t || \{X_i^*\}_{1 \leq i \leq n} || Y || \{Z_j\}_{1 \leq j \leq n} || S)$ . Then the server sends its identity  $S$ , the Diffie-Hellman public value  $Y$ ,  $\{Z_j\}_{1 \leq j \leq n}$  and the authenticator  $V_S$  to subgroup  $SG$ .

**Step 3**

**3.1** Each client  $C_i$ , who belongs to  $SG$ , first looks for  $Z_{j=i}$  and computes the Diffie-Hellman key  $K_i$  with  $x_i$ :  $K_i \equiv Y^{x_i}$ . Now, client  $C_i$  extracts  $S_i$  from  $Z_i$  in an obvious way:  $S_i = Z_i \oplus \mathcal{G}(i, K_i)$ .

**3.2** By collaborating with one another, subgroup  $SG$  reconstructs  $S'$  from the  $t$  shares  $S_i$  by Lagrange interpolation:  $S' = \sum_{k=1}^t \lambda_k \cdot S_k$  where  $\lambda_k \equiv \prod_{1 \leq m \leq t, m \neq k} \frac{m}{m-k}$ . If the received  $V_S$  is not valid (i.e.,  $V_S \neq \mathcal{H}_1(C || S || t || \{X_i^*\}_{1 \leq i \leq n} || Y || \{Z_j\}_{1 \leq j \leq n} || S')$ ), the subgroup terminates the protocol. Otherwise, subgroup  $SG$  generates its session key  $SK \leftarrow \mathcal{H}_2(C || S || t || \{X_i^*\}_{1 \leq i \leq n} || Y || \{Z_j\}_{1 \leq j \leq n} || S')$ . Obviously, any subgroup of less than  $t$  clients cannot generate a common session key  $SK$ .

Instead of collaborating with one another at Step 1.2 and 3.2, one client in the subgroup  $SG$  can choose  $X_j^*$  and reconstruct  $S'$  by collecting  $t$  shares from the other  $t - 1$  clients.

**Remark 1.** *In order to provide mutual authentication in the above protocol, we can simply add the subgroup's authenticator  $V_{SG} \leftarrow \mathcal{H}_3(C\|S\|t\|\{X_i^*\}_{1 \leq i \leq n}\|Y\|\{Z_j\}_{1 \leq j \leq n}\|S)$ , as the third flow from subgroup  $SG$  to server  $S$ , before completing the TAP protocol. This is due to the well-known fact that the basic approach in the literature for adding authentication to an AKE protocol is to use the shared Diffie-Hellman key to construct a simple "authenticator" for the other party [5,3].*

## 5 Security

At first, we show that the TAP protocol of Fig. 2 distributes session keys that are semantically-secure and provides unilateral authentication of server  $S$  in the random oracle model [6].

**Theorem 1. (AKE/UA Security)** *Let  $P$  be the TAP protocol of Fig. 2 where passwords are independently chosen from a dictionary of size  $N$  and  $n$  is the number of clients such that  $n \leq 2\sqrt{N-1} - 1$ .<sup>3</sup> For any adversary  $\mathcal{A}$  within a polynomial time  $t_1$ , with less than  $q_s$  active interactions with the parties (Send-queries),  $q_e$  passive eavesdroppings (Execute-queries) and asking  $q_f$  (resp.,  $q_g$ ) hash queries to  $\mathcal{F}$  (resp.,  $\mathcal{G}$ ),  $\text{Adv}_P^{\text{ake}}(\mathcal{A}) \leq 4\epsilon$  and  $\text{Adv}_P^{\text{S-auth}}(\mathcal{A}) \leq \epsilon$ , with  $\epsilon$  upper-bounded by*

$$\frac{3q_s}{N} + \frac{3nq_g^2}{2} \times \text{Succ}_{g, \mathbb{G}_p}^{\text{cdh}}(t_1 + 3\tau_e) + \frac{q_s}{2^{l_1}} + \frac{(q_e + q_s)^2}{|\mathbb{G}_p|^2} + \frac{q_f^2}{2p} + \frac{q_g^2 + 2(q_s + q_e)}{2|\mathbb{G}_p|},$$

where  $l_1$  is the output length of  $\mathcal{H}_1$  and  $\tau_e$  denotes the computational time for an exponentiation in  $\mathbb{G}_p$ .

This theorem shows that the TAP protocol is secure against off-line dictionary attacks since the advantage of the adversary essentially grows with the ratio of interactions (number of Send-queries) to the number of passwords when  $n \leq 2\sqrt{N-1} - 1$ . We can easily see that the adversary gets a factor  $n$  in the second term since the server generates only one Diffie-Hellman public value and its exponent is used to compute all of the Diffie-Hellman keys  $K_j$ .<sup>4</sup> Due to the lack of space, we leave the proof in the full version of this paper.

Next we prove that the TAP protocol provides client's anonymity against a passive server.

**Theorem 2.** *The TAP protocol provides client's anonymity against a passive server in an information-theoretic sense.*

<sup>3</sup> In practice,  $N = 2^{37}$  for MS-Windows passwords. It is sufficiently large for  $n$ .

<sup>4</sup> If we allow an adversary to corrupt up to  $t - 1$  clients, the factor  $n$  in the security result becomes 1.

*Proof.* Consider server  $S$  who follows the protocol honestly, but it is curious about clients' identities (in  $SG$ ) involved with the TAP protocol. It is obvious that server  $S$  cannot get any information about  $SG$ 's identities since, for each  $i$  ( $1 \leq i \leq n$ ), the  $X_i^*$  has a unique discrete logarithm of  $g$  and, with the randomly chosen  $x_i$ , it is the uniform distribution over  $\mathbb{G}_p$ . This also implies that the server cannot distinguish  $X_i^*$  (of  $C_i \in SG$ ) from  $X_j^*$  (of  $C_j \in C \setminus SG$ ) since they are completely independent one another. In addition, (even if server  $S$  receives the subgroup's authenticator  $V_{SG} \leftarrow \mathcal{H}_3(C||S||t||\{X_i^*\}_{1 \leq i \leq n}||Y||\{Z_j\}_{1 \leq j \leq n}||S)$  at the end of the TAP protocol (in the case of mutual authentication), the  $\{X_i^*\}_{1 \leq i \leq n}$  does not reveal any information about  $SG$ 's identities from the fact that the probability for any subgroup, consisting of  $t$  or more than  $t$  clients, to compute  $S$  is equal.  $\square$

### 6 The Condition on $n$

Here we deduce the condition on  $n$ , appeared in Theorem 1, which is crucial in order to make the security result more meaningful. First, we give an informal definition of security against on-line dictionary attacks: a protocol is said to be secure against on-line dictionary attacks if an adversary can do no better than guess a password during each Send-query (i.e., an impersonation attack). However, the success probability of on-line attacks in the TAP protocol is greater than that in the 2-party PAKE protocols (see below).

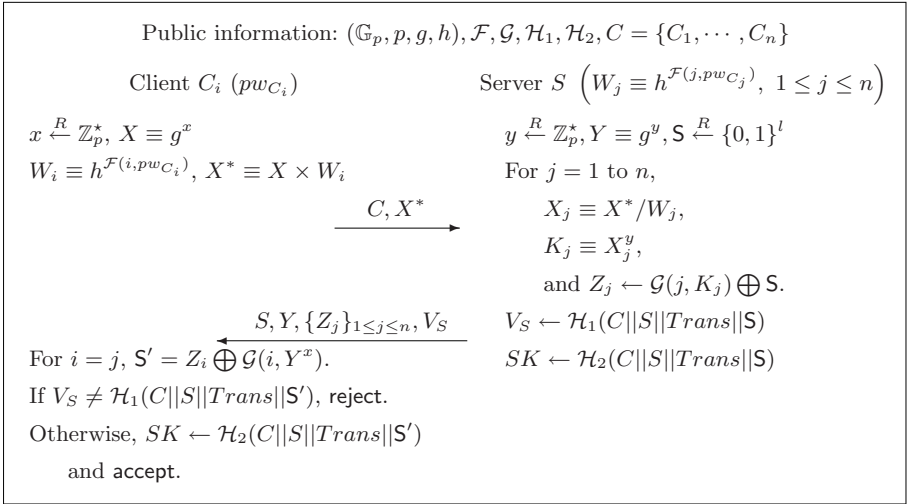
**Theorem 3.** *Consider an adversary who impersonates one party (i.e., subgroup  $SG$  or server  $S$ ) for on-line dictionary attacks in the TAP protocol. Then the probability of the adversary is upper-bounded by*

$$\left\lceil \frac{n}{2} \right\rceil^2 \frac{1}{N(N-1)} .$$

*Proof.* When an adversary invokes Send-queries at **Game  $G_5$**  in the proof, we explain why the probability of on-line dictionary attacks is upper-bounded by the above. In order to maximize  $\Pr[\text{AskH1-WithSG}_5]$ , the strategy the adversary can take is to first determine the threshold  $t$  and guess  $t$  passwords, each of which should be a password of one of  $n/t$  clients. Then the adversary sends the  $t$  and  $\{X_i^*\}_{1 \leq i \leq n}$ , as an honest party  $SG$  would do, to server  $S$ . After receiving the message from the server, the adversary can check whether the guessed passwords are correct or not by seeing the authenticator  $V_S$ . The maximal probability can be obtained when  $t = 2$ . That one password is correct with respect to  $n/2$  clients happens with probability of  $n/2N$ . On the other hand, the probability for the other password is  $n/2(N-1)$ . For any  $n$ , one can get the upper-bound as above since the probability becomes smaller as  $t$  grows. As for  $\Pr[\text{AskH1-WithS}_5]$ , the same discussion can be applied.  $\square$

Now the condition on  $n$  can be easily obtained by restricting the probability of Theorem 3 to  $1/N$ :

$$\left\lceil \frac{n}{2} \right\rceil^2 \frac{1}{N(N-1)} \leq \frac{(n+1)^2}{4N(N-1)} \leq \frac{1}{N} .$$



**Fig. 3.** An efficient anonymous PAKE protocol when  $t = 1$  and where  $Trans = X^* || Y || \{Z_j\}_{1 \leq j \leq n}$

## 7 When the Threshold $t = 1$

If we only consider a passive server in an anonymous PAKE protocol, an efficient construction for the threshold  $t = 1$  can be easily derived from the TAP protocol (see Fig. 3). The main modification from the TAP protocol is that client  $C_i$  only computes his masked Diffie-Hellman public value  $X^*$  and the hash function  $\mathcal{G}$  has the range of  $\{0, 1\}^l$ . In fact, the resultant protocol can be seen as another OT protocol for the case that all the messages of sender is the same. By following the security proof, we can remove the condition on  $n$  because the on-line attacks at **Game  $G_5$**  is limited to one client.

Here, we show how much our protocol of Fig. 3 is efficient compared to the original (not threshold) anonymous PAKE protocol (in Section 3.2 of [20]) in terms of computation costs and communication bandwidth to be required (see Table 1 and 2). In general, the number of modular exponentiations is a major factor to evaluate efficiency of a cryptographic protocol because that is the most power-consuming operation. So we count the number of modular exponentiations as computation costs of client  $C_i$  and server  $S$ . The figures in the parentheses are the remaining number of modular exponentiations after excluding those that are pre-computable. In terms of communication bandwidth,  $|\cdot|$  indicates its bit-length and hash denotes hash functions.

With respect to computation costs in our protocol, client  $C_i$  (resp., server  $S$ ) is required to compute 3 (resp.,  $n + 1$ ) modular exponentiations. When pre-computation is allowed, the remaining costs of client  $C_i$  (resp., server  $S$ ) are 2 (resp.,  $n$ ) modular exponentiations. One can easily see that our protocol has more than 50% reduction from the APAKE protocol in the number of modu-

**Table 1.** Comparison of anonymous PAKE protocols as for computation costs where  $n$  is the number of clients

Protocols	The number of modular exponentiations	
	Client $C_i$	Server $S$
APAKE [20]	6 (4)	$4n + 2$ ( $3n + 1$ )
Our protocol of Fig. 3	3 (2)	$n + 1$ ( $n$ )

**Table 2.** Comparison of anonymous PAKE protocols as for communication bandwidth where  $n$  is the number of clients

Protocols	Communication bandwidth
APAKE [20]	$ C  +  S  + (n + 1) \text{hash}  + (n + 2) p $
Our protocol of Fig. 3	$ C  +  S  + (n + 1) \text{hash}  + 2 p $

lar exponentiations for both client and server. With respect to communication bandwidth, our protocol requires a bandwidth of  $((n + 1)|\text{hash}| + 2|p|)$ -bits except the length of identities  $C$  and  $S$  where the bandwidth for the modulus size  $|p|$  is independent from the number of clients while the APAKE protocol is not. Let us consider the minimum security parameters recommended in practice ( $|p| = 1024$  and  $|\text{hash}| = 160$ ). The gap of communication bandwidths between our and APAKE protocols becomes larger as the number of clients increases.

## 8 Conclusions

After showing insecurity of the previous threshold anonymous PAKE protocol, we have proposed a secure construction (the TAP protocol) which provides not only semantic security of session keys but also anonymity against a passive server. We also proved its security of the TAP protocol in the random oracle model with the reduction to the computational Diffie-Hellman problem. Moreover, we showed the condition on  $n$  in order to get the optimal security result against on-line dictionary attacks. For the threshold  $t = 1$ , we have proposed an efficient anonymous PAKE protocol that can be obtained by slightly modifying the TAP protocol. The resultant protocol significantly improves efficiency in terms of computation costs and communication bandwidth compared to the original (not threshold) anonymous PAKE protocol [20].

## References

1. Abdalla, M., Pointcheval, D.: Simple Password-Based Encrypted Key Exchange Protocols. In: Menezes, A.J. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005)
2. Bellare, M., Canetti, R., Krawczyk, H.: A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In: Proc. of 30th ACM Symposium on Theory of Computing (STOC), pp. 419–428. ACM Press, New York (1998)

3. Bresson, E., Chevassut, O., Pointcheval, D.: New Security Results on Encrypted Key Exchange. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 145–158. Springer, Heidelberg (2004)
4. Bellare, S.M., Merritt, M.: Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In: Proc. of IEEE Symposium on Security and Privacy, pp. 72–84 (1992)
5. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
6. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: Proc. of ACM CCS 1993, pp. 62–73 (1993)
7. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
8. Chu, C.K., Tzeng, W.G.: Efficient  $k$ -Out-of- $n$  Oblivious Transfer Schemes with Adaptive and Non-adaptive Queries. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 172–183. Springer, Heidelberg (2005)
9. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory IT-22(6), 644–654 (1976)
10. Diffie, W., van Oorschot, P., Wiener, M.: Authentication and Authenticated Key Exchange. In: Proc. of Designs, Codes, and Cryptography, pp. 107–125 (1992)
11. <http://grouper.ieee.org/groups/1363/passwdPK/submissions.html>
12. Krawczyk, H.: SIGMA: the 'SIGn-and-MAC' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003)
13. MacKenzie, P., Shrimpton, T., Jakobsson, M.: Threshold Password-Authenticated Key Exchange. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 385–400. Springer, Heidelberg (2002)
14. Nguyen, M.H.: The Relationship Between Password-Authenticated Key Exchange and Other Cryptographic Primitives. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 457–475. Springer, Heidelberg (2005)
15. Shamir, A.: How to Share a Secret. Proc. of Communications of the ACM 22(11), 612–613 (1979)
16. Shoup, V.: On Formal Models for Secure Key Exchange. IBM Research Report RZ 3121, (1999) Available at <http://eprint.iacr.org/1999/012>
17. Shoup, V.: OAEP Reconsidered. Journal of Cryptology 15(4), 223–249 (2002)
18. Shoup, V.: Sequences of Games: A Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive: Report 2004 /332, available at <http://eprint.iacr.org/2004/332>
19. Tzeng, W.G.: Efficient 1-Out- $n$  Oblivious Transfer Schemes. In: Naccache, D., Pailier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 159–171. Springer, Heidelberg (2002)
20. Viet, D.Q., Yamamura, A., Tanaka, H.: Anonymous Password-Based Authenticated Key Exchange. In: Maitra, S., Madhavan, C.E.V., Venkatesan, R. (eds.) INDOCRYPT 2005. LNCS, vol. 3797, pp. 244–257. Springer, Heidelberg (2005)
21. Viet, D.Q., Yamamura, A., Tanaka, H.: Anonymous Password-Based Authenticated Key Exchange. In: Proc. of the 2006 Symposium on Cryptography and Information Security (SCIS 2006), 3D3-4 (January 2006)
22. Wilson, S.B., Johnson, D., Menezes, A.: Key Agreement Protocols and their Security Analysis. In: Proc. of IMA International Conference on Cryptography and Coding (December 1997)