PAPER Special Section on Information and Communication System Security

Threshold Anonymous Password-Authenticated Key Exchange Secure against Insider Attacks

SeongHan SHIN[†,††a)], ***Nonmember***, **Kazukuni KOBARA**[†,††], ***Member***, ***and*** **Hideki IMAI**[†,††], ***Fellow***

SUMMARY An anonymous password-authenticated key exchange (PAKE) protocol is designed to provide both password-only authentication and client anonymity against a semi-honest server, who honestly follows the protocol. In INDOCRYPT2008, Yang and Zhang [26] proposed a new anonymous PAKE (NAPAKE) protocol and its threshold (D-NAPAKE) which they claimed to be secure against insider attacks. In this paper, we first show that the D-NAPAKE protocol [26] is completely insecure against insider attacks unlike their claim. Specifically, only one legitimate client can freely impersonate any subgroup of clients (the threshold $t > 1$) to the server. After giving a security model that captures insider attacks, we propose a threshold anonymous PAKE (called, TAP^+) protocol which provides security against insider attacks. Moreover, we prove that the TAP^+ protocol has semantic security of session keys against active attacks as well as insider attacks under the computational Diffie-Hellman problem, and provides client anonymity against a semi-honest server, who honestly follows the protocol. Finally, several discussions are followed: 1) We also show another threshold anonymous PAKE protocol by applying our Rationale to the non-threshold anonymous PAKE (VEAP) protocol [23]; and 2) We give the efficiency comparison, security consideration and implementation issue of the TAP^+ protocol.

key words: *password-authenticated key exchange, passwords, on-line/off-line dictionary attacks, anonymity, insider attacks, provable security*

1. Introduction

In order to establish secure channels between parties, one of the ways is to use an authenticated key exchange (AKE) protocol that allows the involving parties to authenticate each other and then share an authenticated session key to be used for subsequent cryptographic algorithms (e.g., AES-CBC or MAC). Note that the Diffie-Hellman protocol [8] is a key exchange that does not provide authentication at all.

Adding authentication to a (plain) key exchange protocol is crucial because it can prevent an active adversary, who can completely control the communications, from getting any useful information about the session key. In particular, human-memorable passwords (e.g., 4-digit pin-code or alphanumerical passwords) are commonly used for authentication rather than high-entropy keys because of their convenience in use. Many password-based AKE protocols (see [11] for the exclusive list) 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 very careful about two attacks on passwords: on-line and off-line dictionary attacks. Let us take for example a simple challenge-response password authentication protocol [21] where a client and a server share a password pw. In the protocol, the server sends a challenge c to the client, who computes a response $r = \mathcal{H}(c, pw)$ and sends back r to the server where $\mathcal{H}$ is a hash function. After receiving r, the server authenticates the client if the received r is equal to its own computation $\mathcal{H}(c, pw)$. The on-line dictionary attacks are performed by an adversary who impersonates one party (i.e., the client in the above example) so that the adversary can sieve out possible password candidates one by one. On the other hand, the off-line dictionary attacks are performed off-line and in parallel where an adversary exhaustively enumerates all possible password candidates, in an attempt to determine the correct one, by simply guessing a password and verifying that with additional information. In the above example, an adversary can find out the correct password pw with off-line dictionary attacks by trying all password candidates pw' until it satisfies $r = \mathcal{H}(c, pw')$. While on-line attacks are applicable to all of the password-based protocols equally, they can be prevented by having a server take appropriate countermeasures (e.g., lock-up accounts for 10 minutes after 3 consecutive failures of passwords). But, we cannot avoid off-line attacks by such countermeasures mainly because these attacks can be done off-line and independently of the party.

1.1 Anonymous Password-Authenticated Key Exchange and Its Threshold Version

Quite interestingly, it is not trivial at all to design a secure password-based AKE protocol against off-line dictionary attacks where a client remembers his/her password only and the counterpart server has password verification data. This problem is first discussed in [2] by Bellovin and Merritt, who have also proposed several password-only AKE (called, Encrypted Key Exchange) protocols. Though some protocols turned out to be insecure, their main idea [2] deserves to be reconsidered that by correctly combining symmetric and asymmetric cryptographic techniques we can prevent an adversary from verifying a guessed password (i.e., doing off-line dictionary attacks). Since then, their Encrypted Key Exchange protocols have formed the ba-

Manuscript received February 1, 2011.
Manuscript revised June 3, 2011.

[†]The authors are with the Research Center for Information Security (RCIS), National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba-shi, 305–8568 Japan.
[††]The authors are with the Chuo University, Tokyo, 112–8551 Japan.
a) E-mail: seonghan.shin@aist.go.jp

DOI: 10.1587/transinf.E94.D.2095

Table 1 Previous anonymous PAKE and threshold anonymous PAKE protocols where t is the threshold.

References	Anonymous PAKE ($t = 1$)	Threshold Anonymous PAKE ($t > 1$)
[25]	APAKE	t-out-of-n APAKE (insecure against active attacks)
[22]	TAP ($t = 1$)	TAP ($t > 1$) (secure against active attacks) (insecure against insider attacks)
[26]	NAPAKE	D-NAPAKE (claimed to be secure against insider attacks)
[23]	VEAP (the most efficient)	–

Table 2 Summary of our works where t is the threshold.

	Anonymous PAKE ($t = 1$)	Threshold Anonymous PAKE ($t > 1$)
Our Works	–	D-NAPAKE (insecure against insider attacks [Section 2.2])
		Formal model and definition [Section 3.1]
		TAP$^+$ [Section 4] (secure against active and insider attacks)
		ThresholdVEAP [Section 5.1] obtained by applying our RATIONALE to VEAP [23] (secure against active and insider attacks)

sis for what we call Password-Authenticated Key Exchange (PAKE) protocols. Such protocols [10] have been in standardization of IEEE P1363.2.

In PAKE protocols, a client should send his/her identity clearly in order to mutually authenticate with a server and share a secret that may be the Diffie-Hellman key to be used for generating authenticators and a session key. 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). This information may reflect the client's life pattern and sometimes can be used for spam mails. For this problem, Viet et al., [25] proposed an anonymous PAKE (APAKE) protocol and its threshold construction[†] (t-out-of-n APAKE) both of which simply combine a PAKE protocol [1] for generating secure channels with an Oblivious Transfer (OT) protocol [7], [24] for client's anonymity. The client 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 the client involved with the protocol. In [22], Shin et al., pointed out that the t-out-of-n APAKE protocol [25] is insecure against an outside adversary (i.e., doing off-line dictionary attacks). Also, they proposed an anonymous PAKE (TAP ($t = 1$)) protocol and its threshold (TAP ($t > 1$)) which are only based on the PAKE protocol [1], and showed that their protocols are secure against an outside adversary. In [26], Yang and Zhang first showed that the TAP ($t > 1$) protocol is insecure against an inside adversary and then proposed a new anonymous PAKE (NAPAKE) protocol and its threshold (D-NAPAKE). Their protocols are based on a different PAKE protocol (called, SPEKE [12]–[14]). Recently, Shin et al., [23] proposed an anonymous PAKE (VEAP) protocol that provides the most efficiency in terms of computation and communication costs. Unlike the previous ones, the VEAP protocol is constructed from the blind signature scheme [3], [5]. We summarized the previous works in Table 1.

As a possible application of the (threshold) anonymous PAKE protocols, one can think of the server's public bulletin board on which a message can be posted in an anonymously-authenticated way. After running the (threshold) anonymous PAKE protocols, any subgroup member can post his/her messages securely and anonymously, and also post other's messages because all the subgroup members share the same session key with the server (in the case of threshold anonymous PAKE).

1.2 Our Contributions

The contributions of this paper are as follows (and summarized in Table 2):

- After analyzing the D-NAPAKE protocol [26], we show that it is insecure against insider attacks unlike their claim. Specifically, only one legitimate client can freely impersonate any subgroup of clients (the threshold $t > 1$) to the server.
- In order to capture insider attacks in threshold anonymous PAKE protocols, we give a formal model where an adversary can control less than the threshold number of clients by invoking the Register-query. Also, we

[†]In the threshold construction, the "threshold" number of clients (i.e., a subgroup of the whole clients' group) should collaborate with one another in order to be authenticated by the server. In a different context, MacKenzie et al., [15], [17] proposed a threshold PAKE protocol where the "threshold" number of servers collaborate with one another to resist against compromise of the password verification data. However, such collaborations in the former (resp., latter) protocol require secure channels among the involved clients (resp., servers).

propose a threshold anonymous PAKE (called, TAP^+) protocol that provides security against insider attacks. Moreover, we prove that the TAP^+ protocol is AKE-secure (semantic security of session keys) against active attacks as well as insider attacks under the computational Diffie-Hellman problem in the random oracle model, and provides client anonymity against a semi-honest server, who honestly follows the protocol.

- Finally, several discussions are entailed: 1) We also show another threshold anonymous PAKE protocol by applying our Rationale to the (non-threshold) anonymous PAKE (VEAP) protocol [23]; and 2) We give the efficiency comparison, security consideration and implementation issue of the TAP^+ protocol.

1.3 Organization

This paper is organized as follows. In the next section, we point out that the D-NAPAKE protocol [26] is insecure against insider attacks. In Sect. 3, we give a formal model and security definitions for threshold anonymous PAKE protocols. In Sect. 4, we propose a threshold anonymous PAKE (called, TAP^+) protocol that provides security against insider attacks with its security proof. We give several discussions related to the threshold anonymous PAKE protocols in Sect. 5. Finally, we conclude this paper in Sect. 6.

1.4 Notation

In this subsection, we explain some notation to be used throughout this paper (except Sect. 2). Let $\mathbb{G}_p$ be a finite, cyclic group of prime order p and g be a generator of $\mathbb{G}_p$, where the operation is denoted multiplicatively. Let h be another generator of $\mathbb{G}_p$ such that its discrete logarithm problem with g (i.e., computing $b = \log_g h$) should be hard. This parameter $(\mathbb{G}_p, p, g, h)$ is public to everyone. In the aftermath, all the subsequent arithmetic operations are performed in modulo p unless otherwise stated.

Let l_k be the security parameter for hash functions (i.e., the size of the hashed value). Let $\{0,1\}^*$ be the set of finite binary strings and $\{0,1\}^{l_k}$ be the set of binary strings of length l_k. Let "$\|$" be the concatenation of bit strings in $\{0,1\}^\star$. Let "$\bigoplus$" be the exclusive-OR (XOR) operation of bit strings. If D is a set, then $d \xleftarrow{R} D$ indicates the process of selecting d at random and uniformly over D. If D is a function (whatever it is), then $d \leftarrow D$ indicates the process of assigning the result to d. We denote by $\mathcal{G}, \mathcal{G}_1, \mathcal{G}_2$ full-domain hash (FDH) functions where $\mathcal{G} : \{0,1\}^\star \to \mathbb{Z}_p^\star$ and $\mathcal{G}_1, \mathcal{G}_2 : \{0,1\}^\star \to \mathbb{G}_p$, respectively. The other hash functions are denoted $\mathcal{H}_k : \{0,1\}^\star \to \{0,1\}^{l_k}$, for $k = 1,2,3,4$. Let $C = \{C_1, C_2, \cdots, C_n\}$ and S be the identities of a group of n clients and server, respectively, with each ID $\in \{0,1\}^\star$.

2. Insider Attacks on Previous Threshold Anonymous PAKE Protocol

As explained in the Introduction, a threshold anonymous PAKE protocol allows only the threshold number of clients to authenticate with the corresponding server anonymously. In this section, we show that the D-NAPAKE protocol [26] is not secure against an insider attack where only one legitimate client can freely impersonate any subgroup of clients to the server. In other words, the D-NAPAKE protocol is **NOT** a threshold anonymous PAKE protocol unlike their claim [26].

2.1 The D-NAPAKE Protocol

First, we describe the D-NAPAKE protocol [26] that is designed for any subgroup of clients (denoted by SG) to authenticate with the server (denoted by S) anonymously. The main idea of the D-NAPAKE protocol is that each client belonging to the subgroup and the server share a Diffie-Hellman-like key, by using the SPEKE protocol [12]–[14], and then the subgroup and the server run a sequential Diffie-Hellman protocol, partly-masked with each key, in a threshold secret sharing manner [18]. For the visual description, see Fig. 1.

Let $\mathbb{G} = \langle g \rangle$ be a finite, cyclic group of prime order q, and g be a generator. Let $\mathcal{G} : \{0,1\}^* \to \mathbb{G}$ be a full-domain hash function, and $\mathcal{H}_0, \mathcal{H}_1 : \{0,1\}^* \to \{0,1\}^l$ be two random hash functions where l is the security parameter. Let pw_i be a password shared between the client C_i ($\in \Gamma$) and the server S, and $PW_i = \mathcal{G}(i, pw_i)$. The subgroup SG and the server S agree on the client group $\Gamma = \{C_1, \cdots, C_n\}$ in advance.

1. The server S chooses a random number $r_S \xleftarrow{R} \mathbb{Z}_q^\star$ and, for all n clients in Γ, generates $A_j = PW_j^{r_S}$ where $1 \le j \le n$. Then, server S sends $(S, \{A_j\}_{1 \le j \le n})$ to the subgroup SG.
2. The subgroup $SG \subset \Gamma$ ($|SG| = t$) checks that all the values in $\{A_j\}_{1 \le j \le n}$ are different from one another. If not, subgroup SG aborts the protocol. Otherwise, each client $C_i \in SG$ picks $A_{i=j}$ from $\{A_j\}_{1 \le j \le n}$ and chooses two random numbers $(r_i, x_i) \xleftarrow{R} \mathbb{Z}_q^\star$. Then, each client $C_i \in SG$ computes $X_i = g^{x_i}$, $Z_i = A_i^{r_i}$, $B_{i1} = Z_i \cdot X_i$ and $B_{i2} = PW_i^{r_i}$. The subgroup SG sends $(t, \{B_{i1}, B_{i2}\}_{1 \le i \le t})$ to server S where t ($t \ge 2$) is the threshold.
3. The server S chooses a random number $y \xleftarrow{R} \mathbb{Z}_q^\star$ and computes $Y \equiv g^y$. For j ($1 \le j \le t$), server S generates a share y_j of y, by using Shamir's secret sharing scheme [18] over $\mathbb{Z}_q^\star$, and computes $Z'_j = B_{j2}^{r_S}$, $X'_j = B_{j1}/Z'_j$ and $K_j = (X'_j)^{y_j}$. Also, server S generates an authenticator $Auth_S = \mathcal{H}_1(Trans\|Y)$ and a session key $sk = \mathcal{H}_0(Trans\|Y)$ where $Trans = \Gamma\|S\|\{A_j\}_{1 \le j \le n}\|t\|\{B_{i1}, B_{i2}\}_{1 \le i \le t}\|\{K_j\}_{1 \le j \le t}$. Finally, server S sends $(\{K_j\}_{1 \le j \le t}, Auth_S)$ to subgroup SG.
4. Each client $C_i \in SG$ computes $Y'_i = K_i^{1/x_i}$. Then, the subgroup SG recovers Y' from t Y'_i values by Lagrange interpolation. The subgroup SG checks whether $Auth_S$ is equal to $\mathcal{H}_1(Trans\|Y')$. If not, subgroup SG aborts the protocol. Otherwise, subgroup SG computes a ses-

$\Gamma = \{C_1, \cdots, C_n\}$, $PW_i = \mathcal{G}(i, pw_i)$

Subgroup: $SG \subset \Gamma$ ($\lvert SG \rvert = t$)		Server: S
		$r_S \xleftarrow{R} \mathbb{Z}_q^\star$
	$\xleftarrow{S, \{A_j\}_{1 \le j \le n}}$	For $j = 1$ to n, $A_j = PW_j^{r_S}$
For each $C_i \in SG$,		
$(r_i, x_i) \xleftarrow{R} \mathbb{Z}_q^\star$, $X_i = g^{x_i}$, $Z_i = A_i^{r_i}$		
$B_{i1} = Z_i \cdot X_i$, $B_{i2} = PW_i^{r_i}$	$\xrightarrow{t, \{B_{i1}, B_{i2}\}_{1 \le i \le t}}$	$y \xleftarrow{R} \mathbb{Z}_q^\star, Y = g^y$
		$f(x) = \sum_{k=1}^{t-1} u_k \cdot x^k + y \pmod q$
		where $u_k \xleftarrow{R} \mathbb{Z}_q^\star$
		For $j = 1$ to t,
		$y_j = f(j)$, $Z'_j = B_{j2}^{r_S}$,
		$X'_j = B_{j1}/Z'_j$, $K_j = (X'_j)^{y_j}$
	$\xleftarrow{\{K_j\}_{1 \le j \le t}, Auth_S}$	$Auth_S = \mathcal{H}_1(Trans \lVert Y)$
For each $C_i \in SG$, $Y'_i = K_i^{1/x_i}$		
$Y' = \prod_{C_i \in SG} Y_i'^{\lambda_i}$ where		$sk = \mathcal{H}_0(Trans \lVert Y)$
$\lambda_i = \prod_{1 \le m \le t, m \ne i} \frac{m}{m-i} \pmod q$		
Verify $Auth_S = \mathcal{H}_1(Trans \lVert Y')$		
$sk = \mathcal{H}_0(Trans \lVert Y')$		

Fig. 1 The D-NAPAKE protocol [26] where the threshold (i.e., the number of clients belonging to SG) $t \ge 2$ and $Trans = \Gamma \lVert S \lVert \{A_j\}_{1 \le j \le n} \lVert t \lVert \{B_{i1}, B_{i2}\}_{1 \le i \le t} \lVert \{K_j\}_{1 \le j \le t}$.

sion key $sk = \mathcal{H}_0(Trans \lVert Y')$ and accepts it.

2.2 The Attack

Now, we are ready to show an attack on the D-NAPAKE protocol [26]. W.l.o.g., we assume that a legitimate client $C_l \in \Gamma$, who is sharing his/her password pw_l (and thus $PW_l = \mathcal{G}(l, pw_l)$) with server S, is trying to impersonate any subgroup SG ($t \ge 2$) of clients to server S. Note that Yang and Zhang [26] proposed the D-NAPAKE protocol as a threshold anonymous PAKE protocol so that the threshold ($t \ge 2$) number of clients should participate in the protocol.

1. This is the same as step 1 of Sect. 2.1.
2. After receiving $(S, \{A_j\}_{1 \le j \le n})$ from server S, the client $C_l \in \Gamma$ picks $A_{l=j}$ from $\{A_j\}_{1 \le j \le n}$ and chooses $2t$ random numbers $\{(r_i, x_i)\}_{1 \le i \le t} \xleftarrow{R} \mathbb{Z}_q^\star$. For i ($1 \le i \le t$), client C_l computes $X_i = g^{x_i}$, $Z_i = A_l^{r_i}$, $B_{i1} = Z_i \cdot X_i$ and $B_{i2} = PW_l^{r_i}$. The client C_l sends $(t, \{B_{i1}, B_{i2}\}_{1 \le i \le t})$ to server S where t ($t \ge 2$) is the threshold.
3. This is the same as step 3 of Sect. 2.1.
4. After receiving $(\{K_j\}_{1 \le j \le t}, Auth_S)$ from server S, the client $C_l \in \Gamma$ computes $Y'_i = K_i^{1/x_i}$ for i ($1 \le i \le t$). Then, client C_l recovers Y' from t Y'_i values by Lagrange interpolation. Finally, the client C_l shares a session key $sk = \mathcal{H}_0(Trans \lVert Y')$ with the server S because $Y' = Y$.

Correctness of the attack. It is enough to show that $K_j = (g^{x_i})^{y_j}$ for $i = j$:

$$\begin{aligned} K_j = (X'_j)^{y_j} &= \left(\frac{B_{j1}}{Z'_j}\right)^{y_j} = \left(\frac{B_{j1}}{B_{j2}^{r_S}}\right)^{y_j} \\ &= \left(\frac{Z_i \cdot X_i}{(PW_l^{r_i})^{r_S}}\right)^{y_j} = \left(\frac{A_l^{r_i} \cdot g^{x_i}}{(PW_l^{r_i})^{r_S}}\right)^{y_j} \\ &= \left(\frac{(PW_l^{r_S})^{r_i} \cdot g^{x_i}}{(PW_l^{r_i})^{r_S}}\right)^{y_j} = (g^{x_i})^{y_j} \ . \end{aligned} \tag{1}$$

The problem of the D-NAPAKE protocol resides in the fact that all the Diffie-Hellman-like keys $PW_l^{r_S r_i}$, computed from the SPEKE protocol [12]–[14], are determined by one element r_S and thus can be used to (de-)mask the sequential Diffie-Hellman protocol for t shares of Y (i.e., g^{y_j}). One may think of a simple fix to this attack by having server S choose n different random elements r_{Sj}, for j ($1 \le j \le n$), instead of r_S in step 1 of Sect. 2.1. In that case, the D-NAPAKE protocol should be changed significantly and carefully because the server S has to know which element r_{Sj} is used with which pair $\{B_{i1}, B_{i2}\}$. Also, this simple fix makes the D-NAPAKE protocol *inefficient* and does not guarantee security against any other insider attacks.

3. Formal Model

In this section, we give a formal model (based on [23]) and security definitions for threshold anonymous PAKE protocols. The model described below actually captures insider

attacks by allowing an adversary to control less than the threshold number of clients.

3.1 Model

In a threshold anonymous PAKE protocol P, there are $t+1$ parties SG ($\subset C = \{C_1, C_2, \cdots, C_n\}$ and $|SG| = t$) and S where a pair of client C_i and server S share a low-entropy password pw_i, chosen from a small dictionary $\mathbb{D}_{\mathsf{password}}$, for i ($1 \le i \le n$). We fix the cardinality of $\mathbb{D}_{\mathsf{password}}$ to N. 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^{μ} (resp., S^{ν}) where $\mu, \nu \in \mathbb{N}$, or by I in case of any instance. During the protocol execution, an adversary $\mathcal{A}$ has the entire control of networks (and some clients) 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:

- $\mathsf{Execute}(SG^{\mu}, S^{\nu})$: This query models passive attacks, where the adversary gets access to honest executions of P between the instances SG^{μ} and S^{ν} by eavesdropping.
- $\mathsf{Send}(I, m)$: This query models active attacks by having $\mathcal{A}$ send a message m to an instance I. The adversary $\mathcal{A}$ gets back the response I generates in processing m according to the protocol P. A query $\mathsf{Send}(SG^{\mu}, \mathtt{Start})$ initializes the protocol, and then the adversary receives the first message.
- $\mathsf{Reveal}(I)$: This query handles misuse of the session key [9] by any instance I. 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 $\mathcal{A}$.
- $\mathsf{Register}(C_i, pw_i)$: This query handles insider attacks by having $\mathcal{A}$ register a client C_i to server S with a password pw_i (i.e., $C_i \in C$). That means C_i is completely controlled by $\mathcal{A}$.
- $\mathsf{Test}(I)$: This query does not model any attacks, but it is used to define the AKE security (see Definition 2). The Test-query can be asked at most once by the adversary $\mathcal{A}$ and is only available to $\mathcal{A}$ if the instance I is *fresh* (see below). This query is answered as follows: one flips a private coin $b \in \{0, 1\}$, and forwards the corresponding session key SK ($\mathsf{Reveal}(I)$ would output), if $b = 1$, or a random value with the same size except the session key, if $b = 0$.

Definition 1: (Freshness) Let I' be a partnered instance of I[†]. We say that an instance I is *fresh* if the following conditions hold: (1) the instance has computed and accepted a session key; (2) no Reveal-query has been asked by $\mathcal{A}$ to the instance I and its partner instance I'; (3) $\mathsf{Register}(C_i, \star)$-query has been asked by $\mathcal{A}$ at most up to $t-1$ times, for any i, where t is the threshold of the target instance I; and (4) no $\mathsf{Execute}$-query has been asked by $\mathcal{A}$ if there is any $C_i \in SG$, registered by $\mathsf{Register}(C_i, \star)$-query.

Note that the above freshness definition avoids trivial attacks, for example, by invoking $\mathsf{Register}(C_i, \star)$-query t times or by invoking $\mathsf{Execute}$-query with any $C_i \in SG$ (registered by $\mathsf{Register}(C_i, \star)$-query).

3.2 Security Definitions

The adversary $\mathcal{A}$ is provided with random coin tosses and 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 in the context of executing P. The AKE security is defined by the game $\mathbf{Game}^{\mathsf{ake}}(\mathcal{A}, P)$ where the ultimate goal of the adversary is to guess the bit b, involved in the Test-query, by outputting this guess b'. We denote the AKE advantage, by $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$, as the probability that $\mathcal{A}$ can correctly guess the value of b. Formally,

Definition 2: (AKE Security) A protocol P is said to be AKE secure if, when adversary $\mathcal{A}$ asks q_{send} queries to Send oracle and passwords are chosen from a dictionary of size N, the adversary's advantage $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[b = b'] - 1$ in attacking the protocol P is upper-bounded by

$$O(q_{\mathsf{send}}/N) + \varepsilon(\cdot), \tag{2}$$

for some negligible function $\varepsilon(\cdot)$ in the security parameter. The first term represents the fact that the adversary can do no better than guess a password during each query to Send oracle.

As in [22], [23], [25], we consider a semi-honest server S, who honestly follows the protocol P, but it is curious about the involved clients' identities. The client anonymity is defined by the probability distribution of messages in P.

Definition 3: (Anonymity) Let $P(SG, S)$ (resp., $P(\widetilde{SG}, S)$) be the transcript of P between SG (resp., $\widetilde{SG}$) and S. We can say that the protocol P provides client anonymity if, for any two subgroups SG and $\widetilde{SG}$,

$$\mathsf{Dist}[P(SG, S)] = \mathsf{Dist}[P(\widetilde{SG}, S)] \tag{3}$$

where $\mathsf{Dist}[c]$ denotes c's probability distribution.

This security definition means that the server S gets no information about the clients' identities (in SG) by just observing the protocol transcripts.

4. A Threshold Anonymous PAKE Protocol Secure against Insider Attacks

In this section, we propose a threshold anonymous PAKE (called, TAP^+) protocol that provides security against insider attacks. We also show that the TAP^+ protocol guarantees not only AKE security against active attacks as well as insider attacks but also client anonymity against a semi-honest server, who honestly follows the protocol.

[†]Please, refer to Sect. 3 of [4] for the partnering definition.

Subgroup $SG = \{\underbrace{C_1, C_4, \cdots, C_{n-3}}_{t}\}$

Server S $\boxed{(C_j, W_j \equiv h^{\mathsf{w}_j}) \text{ for } 1 \le j \le n}$

$(y, \mathbf{s}) \xleftarrow{R} (\mathbb{Z}_p^\star)^2, Y \equiv g^y$

Each $C_i \in SG$:

$x_i \xleftarrow{R} \mathbb{Z}_p^\star, X_i \equiv g^{x_i}$,

$\mathsf{w}_i \leftarrow \mathcal{G}(i, pw_i)$, $W_i \equiv h^{\mathsf{w}_i}$,

and $X_i^* \equiv X_i \times W_i$.

Subgroup SG:

$X_j^* \xleftarrow{R} \mathbb{G}_p$ for $C_j \in C \backslash SG$

$\xrightarrow{C, t, \{X_i^*\}_{1 \le i \le n}}$

$f(x) \equiv \sum_{k=0}^{t-1} u_k \cdot x^k$ where $u_0 = \mathbf{s}$

and $u_k \xleftarrow{R} \mathbb{Z}_p^\star$ $(1 \le k \le t-1)$

For $j = 1$ to n,

$\mathbf{s}_j \leftarrow f(j)$, $X_j \equiv X_j^* / W_j$,

$K_j \equiv (X_j)^y$, $Z_j \leftarrow \mathcal{G}_2(X_j^*, Y, K_j, W_j) \bigoplus g^{\mathbf{s}_j}$

and $V_j \leftarrow \mathcal{H}_1(C_j \| X_j^* \| Y \| Z_j \| K_j \| W_j)$.

$V_S \leftarrow \mathcal{H}_2(C \| S \| \textsc{trans} \| g^{\mathbf{s}})$

$\xleftarrow{S, Y, \{Z_j, V_j\}_{1 \le j \le n}, V_S}$

Each $C_i \in SG$:

Look for $\{Z_{j=i}, V_{j=i}\}$, $K_i \equiv Y^{x_i}$

If $V_i \ne \mathcal{H}_1(C_i \| X_i^* \| Y \| Z_i \| K_i \| W_i)$, reject.

Otherwise, $g^{\mathbf{s}_i} = Z_i \bigoplus \mathcal{G}_2(X_i^*, Y, K_i, W_i)$

and output $g^{\mathbf{s}_i}$.

Subgroup SG:

Compute $g^{\mathbf{s}'}$ from $g^{\mathbf{s}_i}$ where $g^{\mathbf{s}'} \equiv g^{\sum_{k=1}^{t} \lambda_k \cdot \mathbf{s}_k}$

and $\lambda_k \equiv \prod_{1 \le m \le t, m \ne k} \frac{m}{m-k} \pmod{p}$

If $V_S \ne \mathcal{H}_2(C \| S \| \textsc{trans} \| g^{\mathbf{s}'})$, reject.

Otherwise, $V_C \leftarrow \mathcal{H}_3(C \| S \| \textsc{trans} \| g^{\mathbf{s}'})$,

$SK \leftarrow \mathcal{H}_4(C \| S \| \textsc{trans} \| g^{\mathbf{s}'})$

and accept.

$\xrightarrow{V_C}$

If $V_C \ne \mathcal{H}_3(C \| S \| \textsc{trans} \| g^{\mathbf{s}})$, reject.

Otherwise, $SK \leftarrow \mathcal{H}_4(C \| S \| \textsc{trans} \| g^{\mathbf{s}})$

and accept.

Fig. 2 A threshold anonymous PAKE (TAP^+) protocol secure against insider attacks where the threshold $t > 1$ and $\textsc{trans} = t \| \{X_i^*\}_{1 \le i \le n} \| Y \| \{Z_j, V_j\}_{1 \le j \le n}$.

4.1 The TAP^+ Protocol

In the TAP^+ protocol, client group C consists of n clients C_i $(1 \le i \le n)$. For simplicity, we assign the clients consecutive integer i $(1 \le i \le n)$ so that C_i can be regarded as the i-th client of $C = \{C_1, C_2, \cdots, C_n\}$. Here, we assume that each client C_i of the group C has registered his/her password pw_i to server S and the latter stores the password verification data W_i, for $1 \le i \le n$, where $W_i \equiv h^{\mathsf{w}_i}$ and $\mathsf{w}_i \leftarrow \mathcal{G}(i, pw_i)$. We also assume that each client C_i in the subgroup SG is connected with the others via pairwise secure channels. In the TAP^+ protocol, any subgroup SG composed of at least t $(t > 1)$ clients wants to share an authenticated session key with server S anonymously (see Fig. 2). Below are descriptions of the TAP^+ protocol.

Step 1

1.1 Each $C_i \in SG$: Each client C_i, who belongs to the subgroup SG, chooses a random number x_i from $\mathbb{Z}_p^\star$ and computes the Diffie-Hellman public value $X_i \equiv g^{x_i}$. The client C_i also computes the password verification data $W_i \equiv h^{\mathsf{w}_i}$ where $\mathsf{w}_i \leftarrow \mathcal{G}(i, pw_i)$, and (i, pw_i) are the index and the password of C_i, respectively. 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 exponent x_i is kept secret by client C_i.

1.2 Subgroup SG: By collaborating with one another, subgroup SG chooses $X_j^* \xleftarrow{R} \mathbb{G}_p$ for each client C_j $(1 \le j \ne i \le n)$, who belongs to the group C but not to the subgroup SG. Then, the subgroup sends the threshold t and $\{X_i^*\}_{1 \le i \le n}$, to the server,

together with the group C of all clients' identities.

Step 2

2.1 The server S chooses two random numbers $(y, \mathbf{s})$ from $\left(\mathbb{Z}_p^\star\right)^2$ and computes its Diffie-Hellman public value $Y \equiv g^y$. The secret $\mathbf{s}$ is distributed as shares by using Shamir's (t, n) secret sharing scheme [18]. 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 $u_0 = \mathbf{s}$ and coefficients u_k $(1 \le k \le t-1)$ randomly chosen from $\mathbb{Z}_p^\star$.

2.2 For the received X_j^* $(1 \le j \le n)$, server S computes $X_j \equiv X_j^*/W_j$ and the Diffie-Hellman key $K_j \equiv \left(X_j\right)^y$. The Z_j is derived from XORing $g^{\mathbf{s}_j}$ and the hashed output of $\mathcal{G}_2$: $Z_j \leftarrow \mathcal{G}_2(X_j^*, Y, K_j, W_j) \bigoplus g^{\mathbf{s}_j}$ where $\mathbf{s}_j \leftarrow f(j)$. For each client C_j, an authenticator V_j is generated as follows: $V_j \leftarrow \mathcal{H}_1(C_j \| X_j^* \| Y \| Z_j \| K_j \| W_j)$.

2.3 Also, server S generates an authenticator $V_S \leftarrow \mathcal{H}_2(C \| S \| t \| \{X_i^*\}_{1 \le i \le n} \| Y \| \{Z_j, V_j\}_{1 \le j \le n} \| g^{\mathbf{s}})$ for subgroup SG. Then, the server sends its identity S, the Diffie-Hellman public value Y, $\{Z_j, V_j\}_{1 \le j \le n}$ and the authenticator V_S to subgroup SG.

Step 3

3.1 Each $C_i \in SG$: Each client C_i, who belongs to the subgroup SG, first looks for the pair $\{Z_{j=i}, V_{j=i}\}$ and computes the Diffie-Hellman key K_i with x_i: $K_i \equiv Y^{x_i}$. If the received V_i is not valid (i.e., $V_i \neq \mathcal{H}_1(C_i \| X_i^* \| Y \| Z_i \| K_i \| W_i)$), client C_i terminates the protocol (and deletes all temporal secrets as well). Otherwise, the client extracts $g^{\mathbf{s}_i}$ from Z_i in an obvious way (i.e., $g^{\mathbf{s}_i} = Z_i \bigoplus \mathcal{G}_2(X_i^*, Y, K_i, W_i)$), and then outputs $g^{\mathbf{s}_i}$ for the subgroup SG.

3.2 Subgroup SG: By collaborating with one another, subgroup SG reconstructs $g^{\mathbf{s}'}$ from the t shares $g^{\mathbf{s}_i}$ by Lagrange interpolation: $g^{\mathbf{s}'} \equiv g^{\sum_{k=1}^{t} \lambda_k \cdot \mathbf{s}_k}$ where $\lambda_k \equiv \prod_{1 \le m \le t, m \neq k} \frac{m}{m-k} \bmod p$. If the received V_S is not valid (i.e., $V_S \neq \mathcal{H}_2(C \| S \| t \| \{X_i^*\}_{1 \le i \le n} \| Y \| \{Z_j, V_j\}_{1 \le j \le n} \| g^{\mathbf{s}'})$), the subgroup terminates the protocol. Otherwise, subgroup SG generates an authenticator $V_C \leftarrow \mathcal{H}_3(C \| S \| t \| \{X_i^*\}_{1 \le i \le n} \| Y \| \{Z_j, V_j\}_{1 \le j \le n} \| g^{\mathbf{s}'})$ and its session key $SK \leftarrow \mathcal{H}_4(C \| S \| t \| \{X_i^*\}_{1 \le i \le n} \| Y \| \{Z_j, V_j\}_{1 \le j \le n} \| g^{\mathbf{s}'})$. The authenticator V_C is sent to server S.

Step 4

4.1 If the received V_C is not valid (i.e., $V_C \neq \mathcal{H}_3(C \| S \| t \| \{X_i^*\}_{1 \le i \le n} \| Y \| \{Z_j, V_j\}_{1 \le j \le n} \| g^{\mathbf{s}})$), server S terminates the protocol (and deletes all temporal secrets as well). Otherwise, the server generates a session key $SK \leftarrow \mathcal{H}_4(C \| S \| t \| \{X_i^*\}_{1 \le i \le n} \| Y \| \{Z_j, V_j\}_{1 \le j \le n} \| g^{\mathbf{s}})$.

Instead of collaborating with one another, one client in the subgroup SG can choose $(n-t)$ X_j^* in **Step 1.2** and reconstruct $g^{\mathbf{s}'}$ by collecting $(t-1)$ shares $g^{\mathbf{s}_i}$ from the others in **Step 3.2**. The reconstructed $g^{\mathbf{s}'}$ is, of course, shared among the involving clients $C_i \in SG$.

Rationale. To be secure against active attacks as well as insider attacks, main rationale behind the TAP^+ protocol is as follows: 1) In **Step 1**, subgroup SG prepares all necessary values for n clients. In fact, only t masked values X_i^* are computed by the involving clients $C_i \in SG$, and each X_i^* is indistinguishable from a randomly chosen X_j^*. It is important in order to provide client anonymity against a semi-honest server; 2) In **Step 2.1** and **2.2**, server S distributes the secret $g^{\mathbf{s}}$ by Shamir's secret sharing scheme so that n shares $g^{\mathbf{s}_j}$ are obtained. For $1 \le j \le n$, the server de-masks the received X_j^* and computes the corresponding Diffie-Hellman key K_j that is used to transport each share $g^{\mathbf{s}_j}$. Obviously, distributing and recovering the secret $g^{\mathbf{s}}$ in **Step 2.1** and **Step 3.2**, respectively, are the threshold part for the TAP^+ protocol. In particular, recovering $g^{\mathbf{s}}$ in **Step 3.2** guarantees security against active attacks because it happens with negligible probability for an adversary (excepting probability of on-line dictionary attacks on the specific clients[†]); and 3) In order to prevent insider attacks, each client C_i should not reveal any useful information on the password pw_i. That is the reason why server S computes the authenticator V_j for each client C_j in **Step 2.2** and the other one V_S for subgroup SG in **Step 2.3**, respectively. If $V_{j=i}$ is invalid, client C_i terminates the protocol. This only allows for inside adversaries to do on-line dictionary attacks (inevitable in the password-only setting)[††].

4.2 Security

In this subsection, we explain the computational Diffie-Hellman (CDH) problem and then show that, under the CDH problem, the TAP^+ protocol of Sect. 4.1 is provably secure in the random oracle model [6].

4.2.1 Computational Assumption

Here, we explain the computational Diffie-Hellman (CDH) problem the TAP^+ protocol is based on.

Definition 4: (CDH Problem) Let $\mathbb{G}_p = \langle g \rangle$ be a finite cyclic group of prime order p with g as a generator. A (t_1, ε_1)-$\mathsf{CDH}_{g,\mathbb{G}_p}$ attacker is a probabilistic polynomial time (PPT) machine $\mathcal{B}$, running in time t_1, such that its success probability $\mathsf{Succ}_{g,\mathbb{G}_p}^{\mathsf{cdh}}(\mathcal{B})$, given random elements g^x and g^y to output g^{xy}, is greater than ε_1. We denote by $\mathsf{Succ}_{g,\mathbb{G}_p}^{\mathsf{cdh}}(t_1)$ the

[†]Note that a guessed password pw_i' can only be tested with the client index i.

[††]Of course, one can take a general countermeasure to on-line dictionary attacks, for example, by making each party to hold one minute after 3 failed trials of passwords (see Sect. 5.3 for more detailed discussion).

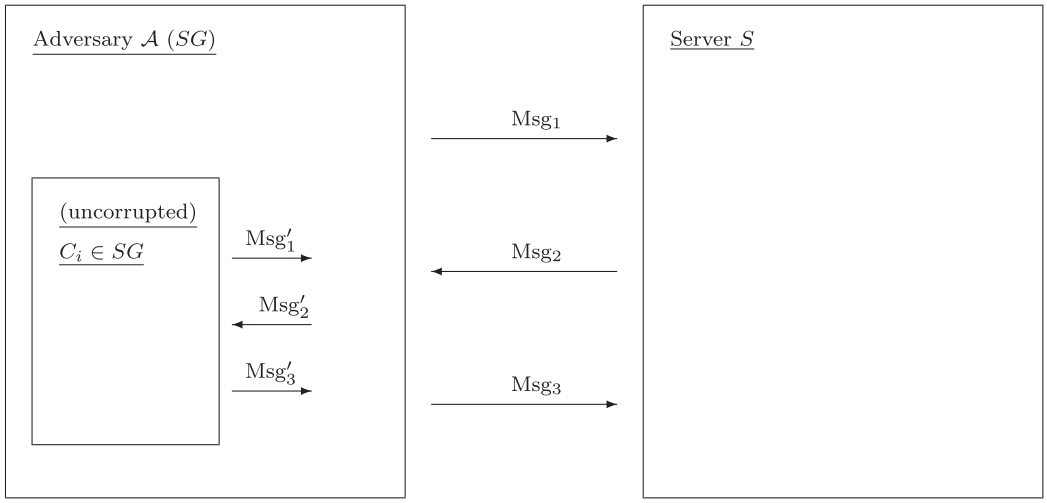


Fig. 3 Simulation abstraction of Theorem 1.

maximal success probability over every adversaries, running within time t_1. The CDH problem states that $\mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1) \leq \varepsilon_1$ for any t_1/ε_1 not too large.

4.2.2 Security Proof

In this subsection, we prove that the TAP^+ protocol is AKE-secure under the CDH problem in the random oracle model [6] and provides unconditional client anonymity against a semi-honest server.

Theorem 1: 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. For any adversary $\mathcal{A}$ within a polynomial time t, with less than q_{send} active interactions with the parties (Send-queries), q_{execute} passive eavesdroppings ($\mathsf{Execute}$-queries) and asking q_{hashG}, q_{hashG2} and q_{hashH} hash queries to $\mathcal{G}$, $\mathcal{G}_2$ and any $\mathcal{H}_k$, for $k = 1, 2, 3, 4$, respectively,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ake}}_P(\mathcal{A}) \leq\ & \frac{2(q_{\mathsf{sendSG}} + nq_{\mathsf{sendS}})}{N} \\ & + 6q^2_{\mathsf{hashG2}} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + 3\tau_e) \\ & + \frac{4nq_{\mathsf{sendS}} + 2q_{\mathsf{hashH2}} + 2Q^2}{|\mathbb{G}_p|} \\ & + \frac{q_{\mathsf{sendSG}}}{2^{l_1+l_2-1}} + \frac{q_{\mathsf{sendS}}}{2^{l_3-1}} + \frac{q^2_{\mathsf{hashH}}}{2^l} \end{aligned} \tag{4}$$

where (1) q_{sendSG} (resp., q_{sendS}) is the number of Send-queries to SG (resp., S) instance, (2) $Q = q_{\mathsf{execute}} + q_{\mathsf{send}} + q_{\mathsf{hashG}} + q_{\mathsf{hashG2}}$, (3) l is the security parameter for the hash functions, (4) l_1, l_2 and l_3 are the output sizes of hash function $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$, respectively, and (5) τ_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 and number of clients to the number of passwords. As it is clear, we have a security loss of factor n in the first term of the security result. The main reason for this loss is that we have to avoid off-line dictionary attacks against an adversary who can corrupt up to $(t-1)$ clients in the subgroup SG ($|SG| = t$) by invoking the $\mathsf{Register}$-queries. In Fig. 3, we depict the simulation abstraction of Theorem 1 where at least one uncorrupted client C_i exists in the subgroup SG and all message exchanges are completely controlled by adversary $\mathcal{A}$. We leave the complete proof of Theorem 1 in Appendix.

Theorem 2: The TAP^+ protocol provides unconditional client anonymity against a semi-honest server.

Proof. Consider server S who honestly follows the TAP^+ protocol, but it is curious about clients' identities (in SG) involved with the protocol. It is obvious that server S cannot get any information about the clients' identities of SG since, for each i ($1 \leq i \leq n$), the X^*_i has a unique discrete logarithm of g and, with the randomly-chosen number 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 $C_j \in C \backslash SG$) since they are completely independent each other. Note that the subgroup's authenticator V_C does not reveal any information about the clients' identities from the fact that the probability, for any subgroup SG and $\widetilde{SG}$ consisting of t or more than t clients, to get the secret $g^{s'}$ is equal. Therefore, $\mathsf{Dist}[P(SG, S)] = \mathsf{Dist}[P(\widetilde{SG}, S)]$ for any two subgroups SG and $\widetilde{SG}$ where $(SG, \widetilde{SG}) \subset C$. □

5. Several Discussions

In this section, we give several discussions related to the threshold anonymous PAKE (including TAP^+) protocols.

5.1 Another Threshold Anonymous PAKE Protocol from VEAP

As a natural extension, we show another threshold anony-

Subgroup $SG = \{\underbrace{C_1, C_4, \cdots, C_{n-3}}_{t}\}$

Server S $\boxed{(C_j, W_j = \mathcal{G}_1(j, pw_j)) \text{ for } 1 \le j \le n}$

Each $C_i \in SG$:

$a_i \xleftarrow{R} \mathbb{Z}_p^\star, A_i \equiv g^{a_i}$,

$W_i \leftarrow \mathcal{G}_1(i, pw_i)$ and $A_i^* \equiv A_i \times W_i$.

[Pre-computation]

$(x, \mathbf{s}) \xleftarrow{R} (\mathbb{Z}_p^\star)^2, X \equiv g^x$

For $j = 1$ to n, $K_j \equiv (W_j)^x$.

Subgroup SG:

$A_j^* \xleftarrow{R} \mathbb{G}_p$ for $C_j \in C \backslash SG$

$\xrightarrow{C, t, \{A_i^*\}_{1 \le i \le n}}$

$f(x) \equiv \sum_{k=0}^{t-1} u_k \cdot x^k$ where $u_0 = \mathbf{s}$

and $u_k \xleftarrow{R} \mathbb{Z}_p^\star$ $(1 \le k \le t-1)$

For $j = 1$ to n,

$\mathbf{s}_j \leftarrow f(j)$, $B_j \equiv \left(A_j^*\right)^x$,

$Z_j \leftarrow \mathcal{G}_2(A_j^*, X, B_j, K_j) \bigoplus g^{\mathbf{s}_j}$

and $V_j \leftarrow \mathcal{H}_1(C_j \| A_j^* \| X \| B_j \| Z_j \| K_j)$.

$V_S \leftarrow \mathcal{H}_2(C \| S \| \text{TRANS} \| g^{\mathbf{s}})$

$\xleftarrow{S, X, \{B_j, Z_j, V_j\}_{1 \le j \le n}, V_S}$

Each $C_i \in SG$:

Look for $\{B_{j=i}, Z_{j=i}, V_{j=i}\}, K_i \equiv B_i / X^{a_i}$

If $V_i \ne \mathcal{H}_1(C_i \| A_i^* \| X \| B_i \| Z_i \| K_i)$, reject.

Otherwise, $g^{\mathbf{s}_i} = Z_i \bigoplus \mathcal{G}_2(A_i^*, X, B_i, K_i)$

and output $g^{\mathbf{s}_i}$.

Subgroup SG:

Compute $g^{\mathbf{s}'}$ from $g^{\mathbf{s}_i}$ where $g^{\mathbf{s}'} \equiv g^{\sum_{k=1}^{t} \lambda_k \cdot \mathbf{s}_k}$

and $\lambda_k \equiv \prod_{1 \le m \le t, m \ne k} \frac{m}{m-k} \pmod{p}$

If $V_S \ne \mathcal{H}_2(C \| S \| \text{TRANS} \| g^{\mathbf{s}'})$, reject.

Otherwise, $V_C \leftarrow \mathcal{H}_3(C \| S \| \text{TRANS} \| g^{\mathbf{s}'})$,

$SK \leftarrow \mathcal{H}_4(C \| S \| \text{TRANS} \| g^{\mathbf{s}'})$

and accept.

$\xrightarrow{V_C}$

If $V_C \ne \mathcal{H}_3(C \| S \| \text{TRANS} \| g^{\mathbf{s}})$, reject.

Otherwise, $SK \leftarrow \mathcal{H}_4(C \| S \| \text{TRANS} \| g^{\mathbf{s}})$

and accept.

Fig. 4 Another threshold anonymous PAKE (ThresholdVEAP) protocol secure against insider attacks where the threshold $t > 1$ and $\text{TRANS} = t \| \{A_i^*\}_{1 \le i \le n} \| X \| \{B_j, Z_j, V_j\}_{1 \le j \le n}$.

mous PAKE (called, ThresholdVEAP) protocol that is secure against insider attacks. The ThresholdVEAP protocol is directly constructed by applying our Rationale of Sect. 4.1 to the (non-threshold) anonymous PAKE (VEAP) protocol [23]. Remember that the VEAP protocol is the most efficient among their kinds in terms of computation and communication costs.

Like the TAP$^+$ protocol of Sect. 4.1, client group C consists of n clients C_i $(1 \le i \le n)$. For simplicity, we assign the clients consecutive integer i $(1 \le i \le n)$ so that C_i can be regarded as the i-th client of $C = \{C_1, C_2, \cdots, C_n\}$. Here, we assume that each client C_i of the group C has registered his/her password pw_i to server S and the latter stores the password verification data W_i, for $1 \le i \le n$, where $W_i = \mathcal{G}_1(i, pw_i)$. In the ThresholdVEAP protocol, any subgroup SG composed of at least t $(t > 1)$ clients wants to share an authenticated session key with server S anonymously (see Fig. 4). Below are descriptions of the ThresholdVEAP protocol.

Step 0 [Pre-computation of server S]

0.1 At first, server S chooses two random numbers $(x, \mathbf{s})$ from $\left(\mathbb{Z}_p^\star\right)^2$ and computes its Diffie-Hellman public value $X \equiv g^x$. For j $(1 \le j \le n)$, the server computes the corresponding key $K_j \equiv \left(W_j\right)^x$ to client C_j.

Step 1

1.1 Each $C_i \in SG$: Each client C_i, who belongs to the subgroup SG, chooses a random number a_i from $\mathbb{Z}_p^\star$ and computes the Diffie-Hellman public value $A_i \equiv g^{a_i}$. The client C_i also computes the password verification data $W_i \leftarrow \mathcal{G}_1(i, pw_i)$, and (i, pw_i) are the index and the password of C_i, respectively. The W_i is used to mask A_i so that its

resultant value A_i^* can be obtained in a way of $A_i^* \equiv A_i \times W_i$. The exponent a_i is kept secret by client C_i.

1.2 Subgroup SG: By collaborating with one another, subgroup SG chooses $A_j^* \overset{R}{\leftarrow} \mathbb{G}_p$ for each client C_j $(1 \leq j \neq i \leq n)$, who belongs to the group C but not to the subgroup SG. Then, the subgroup sends the threshold t and $\{A_i^*\}_{1 \leq i \leq n}$, to the server, together with the group C of all clients' identities.

Step 2

2.1 The server S distributes the secret $\mathbf{s}$ as shares by using Shamir's (t, n) secret sharing scheme [18]. 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 $u_0 = \mathbf{s}$ and coefficients u_k $(1 \leq k \leq t-1)$ randomly chosen from $\mathbb{Z}_p^\star$.

2.2 For the received A_j^* $(1 \leq j \leq n)$, server S computes $B_j \equiv \left(A_j^*\right)^x$ and Z_j. The Z_j is derived from XORing $g^{\mathbf{s}_j}$ and the hashed output of $\mathcal{G}_2$: $Z_j \leftarrow \mathcal{G}_2(A_j^*, X, B_j, K_j) \bigoplus g^{\mathbf{s}_j}$ where $\mathbf{s}_j \leftarrow f(j)$. For each client C_j, an authenticator V_j is generated as follows: $V_j \leftarrow \mathcal{H}_1(C_j \| A_j^* \| X \| B_j \| Z_j \| K_j)$.

2.3 Also, server S generates an authenticator $V_S \leftarrow \mathcal{H}_2(C \| S \| t \| \{A_i^*\}_{1 \leq i \leq n} \| X \| \{B_j, Z_j, V_j\}_{1 \leq j \leq n} \| g^{\mathbf{s}})$ for subgroup SG. Then, the server sends its identity S, the Diffie-Hellman public value X, $\{B_j, Z_j, V_j\}_{1 \leq j \leq n}$ and the authenticator V_S to subgroup SG.

Step 3

3.1 Each $C_i \in SG$: Each client C_i, who belongs to the subgroup SG, first looks for the triplet $\{B_{j=i}, Z_{j=i}, V_{j=i}\}$ and computes the key K_i with a_i: $K_i \equiv B_i / X^{a_i}$. If the received V_i is not valid (i.e., $V_i \neq \mathcal{H}_1(C_i \| A_i^* \| X \| B_i \| Z_i \| K_i)$), client C_i terminates the protocol (and deletes all temporal secrets as well). Otherwise, the client extracts $g^{\mathbf{s}_i}$ from Z_i in an obvious way (i.e., $g^{\mathbf{s}_i} = Z_i \bigoplus \mathcal{G}_2(A_i^*, X, B_i, K_i)$), and then outputs $g^{\mathbf{s}_i}$ for the subgroup SG.

3.2 Subgroup SG: By collaborating with one another, subgroup SG reconstructs $g^{\mathbf{s}'}$ from the t shares $g^{\mathbf{s}_i}$ by Lagrange interpolation: $g^{\mathbf{s}'} \equiv g^{\sum_{k=1}^{t} \lambda_k \cdot \mathbf{s}_k}$ where $\lambda_k \equiv \prod_{1 \leq m \leq t, m \neq k} \frac{m}{m-k} \bmod p$. If the received V_S is not valid (i.e., $V_S \neq \mathcal{H}_2(C \| S \| t \| \{A_i^*\}_{1 \leq i \leq n} \| X \| \{B_j, Z_j, V_j\}_{1 \leq j \leq n} \| g^{\mathbf{s}'})$), the subgroup terminates the protocol. Otherwise, subgroup SG generates an authenticator $V_C \leftarrow \mathcal{H}_3(C \| S \| t \| \{A_i^*\}_{1 \leq i \leq n} \| X \| \{B_j, Z_j, V_j\}_{1 \leq j \leq n} \| g^{\mathbf{s}'})$ and its session key $SK \leftarrow \mathcal{H}_4(C \| S \| t \| \{A_i^*\}_{1 \leq i \leq n} \| X \| \{B_j, Z_j, V_j\}_{1 \leq j \leq n} \| g^{\mathbf{s}'})$. The authenticator V_C is sent to server S.

Step 4

4.1 If the received V_C is not valid (i.e., $V_C \neq \mathcal{H}_3(C \| S \| t \| \{A_i^*\}_{1 \leq i \leq n} \| X \| \{B_j, Z_j, V_j\}_{1 \leq j \leq n} \| g^{\mathbf{s}}))$, server S terminates the protocol (and deletes all temporal secrets as well). Otherwise, the server generates a session key $SK \leftarrow \mathcal{H}_4(C \| S \| t \| \{A_i^*\}_{1 \leq i \leq n} \| X \| \{B_j, Z_j, V_j\}_{1 \leq j \leq n} \| g^{\mathbf{s}})$.

Though the ThresholdVEAP protocol is based on the most efficient anonymous PAKE (VEAP) protocol [23], it does not have advantages in terms of server's computation costs (without pre-computation) and communication costs over the TAP$^+$ protocol of Sect. 4.1. See the next subsection, for more detailed comparison.

Note that the ThresholdVEAP and TAP$^+$ protocols are similar because they follow the same Rationale in Sect. 4.1. However, the actual constructions (including their efficiency) of ThresholdVEAP and TAP$^+$ are different since the former's core primitive is the PAKE protocol [1] and the latter's is the blind signature scheme [3], [5].

5.2 Efficiency Comparison

In this subsection, we show the efficiency comparison between the TAP$^+$ protocol of Sect. 4.1 and the ThresholdVEAP protocol of Sect. 5.1 in terms of computation and communication costs (see Table 3). Note that they are the only secure threshold anonymous PAKE protocols against active attacks as well as insider attacks.

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 each client $C_i \in SG$ and server S. In Table 3, "Total" means the total number of modular exponentiations and "Remaining" is the remaining number of modular exponentiations after excluding those that are pre-computable. In terms of communication costs, $|p|$ and $|\mathcal{H}|$ indicate the bit-length of group order p and of hash function $\mathcal{H}$, respectively.

With respect to computation costs in the TAP$^+$ protocol, each client C_i (resp., server S) is required to compute 3 (resp., $2n + 2$) modular exponentiations. When pre-computation is allowed, the remaining costs of each client C_i (resp., server S) are 2 (resp., $2n$) modular exponentiation. With respect to communication costs, the TAP$^+$ protocol requires a bandwidth of $((2n+1)|p| + (n+2)|\mathcal{H}|)$-bits except the length of identities C and S. From Table 3, we can easily see that the TAP$^+$ protocol is more efficient in terms of server's computation costs (without pre-computation) and communication costs than the ThresholdVEAP protocol.

5.3 Security Consideration

As we showed in Sect. 4.2.2, the AKE security of the TAP$^+$ protocol has a reduction to the computational Diffie-Hellman problem (i.e., weak assumption). However, we get a security loss of factor n in the probability of on-line dictionary attacks (see Eq. (4)).

Table 3 Efficiency comparison of threshold anonymous PAKE protocols, secure against active attacks and insider attacks, where n is the number of clients.

Protocols	Number of modular exponentiations				Communication costs *1
	Each client $C_i \in SG$		Server S		
	Total	Remaining	Total	Remaining	
TAP^+ (Section 4.1)	3	2	$2n+2$	$2n$	$(2n+1)\|p\| + (n+2)\|\mathcal{H}\|$
ThresholdVEAP (Section 5.1)	2	1	$3n+2$	$2n$	$(3n+1)\|p\| + (n+2)\|\mathcal{H}\|$

*1: The bit-length of identities is excluded

If n is relatively small compared to N, the security loss can be not considered significant. For example, $N = 2^{37}$ for MS-Windows passwords. According to [16], it would take about 90 years to carry out $2^{25.5}$ trials with one minute lock out for 3 failed trials. Therefore, if $n \approx 2^{10}$ we can get a reasonable security margin.

If n is large ($n \gg 2^{10}$), the security loss would be in trouble. For that, there may be two possible solutions. The first (somewhat impractical) solution is that each client chooses a longer password from the large password space. The second is to reduce the loss of n to the threshold t by redesigning the TAP^+ protocol. The reason we have such a security loss is that any adversary, who impersonates the subgroup SG, can try n passwords in the first message $\{X_i^*\}_{1\le i\le n}$ and verify these passwords with the authenticators $\{V_j\}_{1\le j\le n}$ of the second message. In order for an adversary to test only t passwords, we have to send t X_i^* and subsequently the server S should respond with $(t\times n)$ Z_j and V_j. Of course, the resultant protocol is quite inefficient and, in fact, we have a trade-off between the security loss and efficiency of the TAP^+ protocol.

5.4 Implementation Issue

In order to reduce computation costs (i.e., modular exponentiations) in the TAP^+ and ThresholdVEAP protocols, one can consider using a finite cyclic subgroup $\mathbb{G}_{p,q}$ of prime order q of the multiplicative group $\mathbb{Z}_p^*$ where $p = aq + 1$ is a prime, a is an integer, and g is a generator of $\mathbb{G}_{p,q}$. In this case, one should be careful to implementing FDH functions $\mathcal{G}, \mathcal{G}_1, \mathcal{G}_2$ and selecting $X_j^* \overset{R}{\leftarrow} \mathbb{G}_{p,q}$ (resp., $A_j^* \overset{R}{\leftarrow} \mathbb{G}_{p,q}$) in the TAP^+ (resp., ThresholdVEAP) protocol since they need additional computation costs for the group membership test.

6. Conclusions

After analyzing the D-NAPAKE protocol [26], we showed that it is completely insecure against insider attacks unlike the authors' claim. Specifically, only one legitimate client can freely impersonate any subgroup of clients (the threshold $t > 1$) to the server. In order to capture insider attacks, we gave a formal model where an adversary can control less than the threshold number of clients by invoking the Register-query. Also, we proposed a threshold anonymous PAKE (called, TAP^+) protocol that provides security against insider attacks. Moreover, we proved that the TAP^+ protocol is AKE-secure against active attacks as well as insider attacks under the computational Diffie-Hellman problem, and provides client anonymity against a semi-honest server, who honestly follows the protocol. Finally, several discussions were followed: 1) We also showed another threshold anonymous PAKE protocol by applying our Rationale to the (non-threshold) anonymous PAKE (VEAP) protocol [23]; and 2) We gave the efficiency comparison, security consideration and implementation issue of the TAP^+ protocol.

As we discussed in Sect. 5.3, an open (and challenging) problem is to design a threshold anonymous PAKE protocol that is secure against insider attacks and has a *tight* security reduction (independent of the number of clients n) in the probability of on-line dictionary attacks. We also leave it as an open issue to deal with potential Denial-of-Service (DoS) attacks to the client group because, if an adversary impersonates any client in a threshold anonymous PAKE protocol, the server cannot identify the client exactly as well as restrict his/her further access.

Acknowledgments

We sincerely appreciate the anonymous reviewers' constructive comments and advices on this paper. The first author was partly supported by JSPS KAKENHI 22760285.

References

[1] M. Abdalla and D. Pointcheval, "Simple password-based encrypted key exchange protocols," Proc. CT-RSA 2005, LNCS 3376, pp.191–208, Springer-Verlag, 2005.
[2] S.M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," Proc. IEEE Symposium on Security and Privacy, pp.72–84, 1992.
[3] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, "The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme," J. Cryptology, vol.16, no.3, pp.185–215, 2003.
[4] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," Proc. EUROCRYPT 2000, LNCS 1807, pp.139–155, Springer-Verlag, 2000.
[5] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," Proc. PKC 2003, LNCS 2567, pp.31–46, Springer-Verlag, 2003. The full version is availabel at http://www.cse.ucsd.edu/users/aboldyre/papers/b.html
[6] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," Proc. ACM CCS'93, pp.62–73, ACM Press, 1993.
[7] C.K. Chu and W.G. Tzeng, "Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries," Proc. PKC 2005, LNCS 3386, pp.172–183, Springer-Verlag, 2005.
[8] W. Diffie and M. Hellman, "New directions in cryptography," IEEE Trans. Inf. Theory, vol.IT-22, no.6, pp.644–654, 1976.

[9] D. Denning and G. Sacco, "Timestamps in key distribution protocols," Commun. ACM, vol.24, pp.533–536, 1981.
[10] IEEE P1363.2, "Standard specifications for password-based public key cryptographic techniques," Available at http://grouper.ieee.org/groups/1363/passwdPK/submissions.html
[11] Research Papers on Password-based Cryptography, http://www.jablon.org/passwordlinks.html
[12] D. Jablon, "Strong password-only authenticated key exchange," ACM Computer Communication Review, vol.26, no.5, pp.5–20, ACM SIGCOMM, 1996.
[13] D. Jablon, "Extended password key exchange protocols immune to dictionary attacks," WET-ICE'97 Workshop on Enterprise Security, 1997.
[14] P. MacKenzie, "On the security of the SPEKE password-authenticated key exchange protocol," Cryptology ePrint Archive: Report 2001/057, Available at http://eprint.iacr.org/2001/057
[15] P. MacKenzie, T. Shrimpton, and M. Jakobsson, "Threshold password-authenticated key exchange," Proc. CRYPTO 2002, LNCS 2442, pp.385–400, Springer-Verlag, 2002.
[16] NIST, "Information security (electronic authentication guideline)," Special Publication 800-63, April 2006. Available at http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf
[17] M.D. Raimondo and R. Gennaro, "Provably secure threshold password-authenticated key exchange," Proc. EUROCRYPT 2003, LNCS 2656, pp.507–523, Springer-Verlag, 2003.
[18] A. Shamir, "How to share a secret," Commun. ACM, vol.22, no.11, pp.612–613, 1979.
[19] V. Shoup, "OAEP reconsidered," J. Cryptology, vol.15, no.4, pp.223–249, 2002.
[20] V. Shoup, "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
[21] W. Simpson, "PPP challenge handshake authentication protocol (CHAP)," IETF RFC 1994, Aug. 1996. Available at http://www.ietf.org/rfc/rfc1994.txt
[22] S.H. Shin, K. Kobara, and H. Imai, "A secure threshold anonymous password-authenticated key exchange protocol," Proc. IWSEC 2007, LNCS 4752, pp.444–458, Springer-Verlag, 2007.
[23] S.H. Shin, K. Kobara, and H. Imai, "Very-efficient anonymous password-authenticated key exchange and its extensions," Proc. AAECC 2009, LNCS 5527, pp.149–158, Springer-Verlag, 2009.
[24] W.G. Tzeng, "Efficient 1-out-n oblivious transfer schemes," Proc. PKC 2002, LNCS 2274, pp.159–171, Springer-Verlag, 2002.
[25] D.Q. Viet, A. Yamamura, and H. Tanaka, "Anonymous password-based authenticated key exchange," Proc. INDOCRYPT 2005, LNCS 3797, pp.244–257, Springer-Verlag, 2005.
[26] J. Yang and Z. Zhang, "A new anonymous password-based authenticated key exchange protocol," Proc. INDOCRYPT 2008, LNCS 5365, pp.200–212, Springer-Verlag, 2008.

Appendix: Proof of Theorem 1

In this proof, we define a sequence of games starting at game $\mathbf{G}_0$ (i.e., the actual TAP^+ protocol) and ending up at $\mathbf{G}_5$ where the hash functions are modelled as random oracles. We use Shoup's difference lemma [19], [20] to bound the probability of each event in these games. Let S_i be an event where an adversary correctly guesses the bit b, involved in the Test-query, in **Game** $\mathbf{G}_i$. For visual simplicity, we denote $\{X_i^*\}_{1\le i\le n}$ and $\{Z_j, V_j\}_{1\le j\le n}$ by $\{X_i^*\}$ and $\{Z_j, V_j\}$, respectively, in the following proof.

Game $\mathbf{G}_0$: This is the real protocol in the random oracle model. By AKE-security definition,

$$\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[\mathsf{S}_0] - 1\ . \qquad (\mathrm{A}\cdot 1)$$

Game $\mathbf{G}_1$: In this game, we simulate the hash oracles ($\mathcal{G}$, $\mathcal{G}_2$ and $\mathcal{H}_k$, but as well additional hash functions $\mathcal{H}'_k : \{0,1\}^\star \to \{0,1\}^{l_k}$, for $k = 1,2,3,4$) as usual by maintaining these hash lists $\Lambda_{\mathcal{G}}$, $\Lambda_{\mathcal{G}2}$, $\Lambda_{\mathcal{H}}$ and $\Lambda_{\mathcal{H}'}$ (see below). We also simulate all the instances, as the real parties would do, for the Send-queries and for the Execute, Reveal, Register and Test-queries (see further below).

Simulation of the hash functions: $\mathcal{G}$, $\mathcal{G}_2$ and $\mathcal{H}_k$

- For a hash-query $\mathcal{G}(q)$, such that a record (q, r) appears in $\Lambda_{\mathcal{G}}$, the answer is r. Otherwise, one chooses a random element $r \xleftarrow{R} \mathbb{Z}_p^\star$, answers with it, and adds the record (q, r) to $\Lambda_{\mathcal{G}}$.
- For a hash-query $\mathcal{G}_2(q)$, such that a record (q, r) appears in $\Lambda_{\mathcal{G}_2}$, the answer is r. Otherwise, one chooses a random element $r \xleftarrow{R} \mathbb{G}_p$, answers with it, and adds the record (q, r) to $\Lambda_{\mathcal{G}_2}$.
- For a hash-query $\mathcal{H}_k(q)$ (resp., $\mathcal{H}'_k(q)$), such that a record (k, q, r) appears in $\Lambda_{\mathcal{H}}$ (resp., $\Lambda_{\mathcal{H}'}$), the answer is r. Otherwise, one chooses a random element $r \xleftarrow{R} \{0,1\}^{l_k}$, answers with it, and adds the record (k, q, r) to $\Lambda_{\mathcal{H}}$ (resp., $\Lambda_{\mathcal{H}'}$).

Simulation of the TAP^+ protocol

Setup

First, we prepare for the public parameter $(\mathbb{G}_p, p, g)$ as usual and generate another generator h as follows:

- For h, we apply the following rule:
  - ► **Rule Setup**$^{(1)}$
    Choose a random element $b \xleftarrow{R} \mathbb{Z}_p^\star$ and compute $h \equiv g^b$.

Send-queries to SG

We answer to the Send-queries to a SG-instance as follows:

- A Send(SG^μ, `Start`)-query is processed by first setting the threshold t ($t > 1$) and randomly selecting t indices from the set C. We apply the following rules:
  - ► **Rule SG1**$^{(1)}$
    For $C_i \in SG$, choose a random element $\theta_i \xleftarrow{R} \mathbb{Z}_p^\star$, and compute $X_i \equiv g^{\theta_i}$, $W_i \equiv h^{\mathsf{w}_i}$ and $X_i^* \equiv X_i \times W_i$ where $\mathsf{w}_i \leftarrow \mathcal{G}(i, pw_i)$.
  - ► **Rule SG2**$^{(1)}$
    For $C_j \in C \backslash SG$, choose a random element $X_j^* \xleftarrow{R} \mathbb{G}_p$.

  Then, the query is answered with $(C, t, \{X_i^*\})$, and the instance goes to an expecting state.

- If the instance SG^{μ} is in an expecting state, a query $\mathsf{Send}(SG^{\mu}, (S, Y, \{Z_j, V_j\}, V_S))$ is processed by reconstructing the secret $g^{\mathbf{s}'}$ and by computing the authenticator and the session key. We apply the following rules.
  - ► **Rule SG3**$^{(1)}$
    For $C_i \in SG$, compute $K_i \equiv Y^{\theta_i}$ and $V_i' \leftarrow \mathcal{H}_1(C_i \| X_i^* \| Y \| Z_i \| K_i \| W_i)$. If $V_i' = V_i$, compute $g^{\mathbf{s}_i} = Z_i \bigoplus \mathcal{G}_2(X_i^*, Y, K_i, W_i)$.

  If $V_i' \neq V_i$, the instance terminates.
  - ► **Rule SG4**$^{(1)}$
    Compute $V_S' \leftarrow \mathcal{H}_2(C \| S \| t \| \{X_i^*\} \| Y \| \{Z_j, V_j\} \| g^{\mathbf{s}'})$. If $V_S' = V_S$, compute the expected authenticator and the session key:
    $V_C \leftarrow \mathcal{H}_3(C \| S \| t \| \{X_i^*\} \| Y \| \{Z_j, V_j\} \| g^{\mathbf{s}'})$,
    $SK_{SG} \leftarrow \mathcal{H}_4(C \| S \| t \| \{X_i^*\} \| Y \| \{Z_j, V_j\} \| g^{\mathbf{s}'})$.

  If $V_S' \neq V_S$, it terminates. Otherwise, the query is answered with V_C, and the instance accepts and terminates.

<u>Send-queries to S</u>

We answer to the Send-queries to a S-instance as follows:

- A $\mathsf{Send}(S^{\nu}, (C, t, \{X_i^*\}))$-query is processed according to the following rule:
  - ► **Rule S1**$^{(1)}$
    Choose two random elements $(\varphi, \mathbf{s}) \xleftarrow{R} \left(\mathbb{Z}_p^{\star}\right)^2$ and compute $Y \equiv g^{\varphi}$.

  Then, the instance computes the authenticators after generating the shares $\mathbf{s}_j$ of $\mathbf{s}$ by (t, n)-threshold secret sharing scheme [18]. We apply the following rules:
  - ► **Rule S2**$^{(1)}$
    For j $(1 \leq j \leq n)$, compute $X_j \equiv X_j^*/W_j$, $K_j \equiv \left(X_j\right)^{\varphi}$ and $Z_j \leftarrow \mathcal{G}_2(X_j^*, Y, K_j, W_j) \bigoplus g^{\mathbf{s}_j}$.
  - ► **Rule S3**$^{(1)}$
    For j $(1 \leq j \leq n)$, compute $V_j \leftarrow \mathcal{H}_1(C_j \| X_j^* \| Y \| Z_j \| K_j \| W_j)$. Also, compute $V_S \leftarrow \mathcal{H}_2(C \| S \| t \| \{X_i^*\} \| Y \| \{Z_j, V_j\} \| g^{\mathbf{s}})$.

  Then, the query is answered with $(S, Y, \{Z_j, V_j\}, V_S)$, and then the instance goes to an expecting state.
- If the instance S^{ν} is in an expecting state, a query $\mathsf{Send}(S^{\nu}, V_C)$ is processed as follows:
  - ► **Rule S4**$^{(1)}$
    Compute $V_C' \leftarrow \mathcal{H}_3(C \| S \| t \| \{X_i^*\} \| Y \| \{Z_j, V_j\} \| g^{\mathbf{s}})$. If $V_C' = V_C$, compute the session key:
    $SK_S \leftarrow \mathcal{H}_4(C \| S \| t \| \{X_i^*\} \| Y \| \{Z_j, V_j\} \| g^{\mathbf{s}})$.

  If $V_C' \neq V_C$, it terminates. Otherwise, the instance accepts and terminates.

<u>Other queries</u>

- An $\mathsf{Execute}(SG^{\mu}, S^{\nu})$-query is processed using successively the above simulations of the Send-queries: $(C, t, \{X_i^*\}) \leftarrow \mathsf{Send}(SG^{\mu}, \mathtt{Start})$, $(S, Y, \{Z_j, V_j\}, V_S) \leftarrow \mathsf{Send}(S^{\nu}, (C, t, \{X_i^*\}))$, $V_C \leftarrow \mathsf{Send}(SG^{\mu}, (S, Y, \{Z_j, V_j\}, V_S))$, and then outputting the transcript $((C, t, \{X_i^*\}), (S, Y, \{Z_j, V_j\}, V_S), V_C)$.
- A $\mathsf{Reveal}(I)$-query returns the session key (SK_{SG} or SK_S) computed by the instance I (if the latter has accepted).
- A $\mathsf{Register}(C_i, pw_i)$-query is processed as follows: compute $W_i \equiv h^{\mathsf{w}_i}$, where $\mathsf{w}_i \leftarrow \mathcal{G}(i, pw_i)$, and stores W_i as the verification data for C_i.
- A $\mathsf{Test}(I)$-query first gets SK from $\mathsf{Reveal}(I)$, and flip a coin b. If $b = 1$, we return the value of the session key SK, otherwise we return a random value drawn from $\{0, 1\}^{l_4}$.

From the above simulation, we can easily see that the game is perfectly indistinguishable from the real attack.

$$\Pr[\mathsf{S}_1] \approx \Pr[\mathsf{S}_0] \,. \tag{A·2}$$

Game $\mathbf{G}_2$: In this game, we consider collisions that is the difference from Game $\mathbf{G}_1$. Let Coll_2 be an event where some collisions are going to happen.

- Collisions on the partial transcripts $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}))$: An adversary tries to find out a pair $((t, \{X_i^*\}), (Y, \{Z_j\}))$, coinciding with the challenge transcript, and then obtain the corresponding session key using the Reveal-query. However, at least one party involves with the transcripts, and thus one of $(t, \{X_i^*\})$ and $(Y, \{Z_j\})$ is truly uniformly distributed. Note that the adversary can control at most $(t-1)$ clients by the Register-query.
- Collisions on the output of $\mathcal{G}$ and $\mathcal{G}_2$
- Collision on the output of $\mathcal{H}$

These probabilities are upper-bounded by the birthday paradox:

$$\Pr[\mathsf{Coll}_2] \leq \frac{(q_{\mathsf{execute}} + q_{\mathsf{send}})^2}{2|\mathbb{G}_p|} \times \left(1 + \frac{1}{|\mathbb{G}_p|}\right) + \frac{q_{\mathsf{hashG}}^2 + q_{\mathsf{hashG2}}^2}{2|\mathbb{G}_p|} + \frac{q_{\mathsf{hashH}}^2}{2^{l+1}} \,. \tag{A·3}$$

Note that the first term is the maximum upper-bound of collisions on the partial transcripts.

Game $\mathbf{G}_3$: In this game, we make the authenticators and the session key unpredictable to any adversary by using the private oracles $\mathcal{H}_k'$ instead of $\mathcal{H}_k$, for $k = 1, 2, 3, 4$. For that, we apply the following rules:

- ► **Rule SG3**$^{(3)}$
  For $C_i \in SG$, compute $K_i \equiv Y^{\theta_i}$ and $V_i' \leftarrow \mathcal{H}_1'(C_i \| X_i^* \| Y \| Z_i)$. If $V_i' = V_i$, compute $g^{\mathbf{s}_i} = Z_i \bigoplus \mathcal{G}_2(X_i^*, Y, K_i, W_i)$.

► **Rule SG4**$^{(3)}$
Compute $V'_S \leftarrow \mathcal{H}'_2(C\|S\|t\|\{X_i^*\}\|Y\|\{Z_j, V_j\})$. If $V'_S = V_S$, compute the expected authenticator and the session key:
$V_C \leftarrow \mathcal{H}'_3(C\|S\|t\|\{X_i^*\}\|Y\|\{Z_j, V_j\})$,
$SK_{SG} \leftarrow \mathcal{H}'_4(C\|S\|t\|\{X_i^*\}\|Y\|\{Z_j, V_j\})$.

► **Rule S3**$^{(3)}$
For j $(1 \le j \le n)$, compute $V_j \leftarrow \mathcal{H}'_1(C_j\|X_j^*\|Y\|Z_j)$. Also, compute $V_S \leftarrow \mathcal{H}'_2(C\|S\|t\|\{X_i^*\}\|Y\|\{Z_j, V_j\})$.

► **Rule S4**$^{(3)}$
Compute $V'_C \leftarrow \mathcal{H}'_3(C\|S\|t\|\{X_i^*\}\|Y\|\{Z_j, V_j\})$. If $V'_C = V_C$, compute the session key:
$SK_S \leftarrow \mathcal{H}'_4(C\|S\|t\|\{X_i^*\}\|Y\|\{Z_j, V_j\})$.

Note that we do not need to use the key K_j and the secret $g^{\mathbf{s}}$ in the computation of $\mathcal{H}'_k$. Accordingly, we can simplify the following rules:

► **Rule S1**$^{(3)}$
Choose a random element $\varphi \overset{R}{\leftarrow} \mathbb{Z}_p^\star$ and compute $Y \equiv g^\varphi$.

► **Rule S2**$^{(3)}$
For j $(1 \le j \le n)$, choose a random element $Z_j \leftarrow \mathbb{G}_p$.

For an *uncorrupted* C_i, we use the homomorphic property of $\mathbb{G}_p$ so that the generation of X_i^* is simplified as follows:

► **Rule SG1**$^{(3)}$
For an uncorrupted $C_i \in SG$, choose a random element $\theta_i \overset{R}{\leftarrow} \mathbb{Z}_p^\star$ and compute $X_i^* \equiv g^{\theta_i}$.

The games $\mathbf{G}_3$ and $\mathbf{G}_2$ are indistinguishable unless some specific hash queries are asked, denoted by event $\mathsf{AskH}_3 = \mathsf{AskH12}_3 \vee \mathsf{AskH3w12}_3 \vee \mathsf{AskH4w123}_3$:

- $\mathsf{AskH12}_3$: $\mathcal{H}_1(C_j\|X_j^*\|Y\|Z_j\|W_j\|K_j)$ and $\mathcal{H}_2(C\|S\|t\|\{X_i^*\}\|Y\|\{Z_j, V_j\}\|g^{\mathbf{s}})$ have been queried by $\mathcal{A}$ to $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, for some execution transcripts $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}))$;
- $\mathsf{AskH3w12}_3$: $\mathcal{H}_3(C\|S\|t\|\{X_i^*\}\|Y\|\{Z_j, V_j\}\|g^{\mathbf{s}})$ has been queried by $\mathcal{A}$ to $\mathcal{H}_3$ for some execution transcripts $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}))$, but event $\mathsf{AskH12}_3$ did not happen;
- $\mathsf{AskH4w123}_3$: $\mathcal{H}_4(C\|S\|t\|\{X_i^*\}\|Y\|\{Z_j, V_j\}\|g^{\mathbf{s}})$ has been queried by $\mathcal{A}$ to $\mathcal{H}_4$ for some execution transcripts $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}))$, where some party has accepted, but events $\mathsf{AskH12}_3$ and $\mathsf{AskH3w12}_3$ did not happen;

The above obviously leads to the following (these probabilities are computed at the Game $\mathbf{G}_5$):

$$\Pr[\mathsf{AskH}_3] \le \Pr[\mathsf{AskH12}_3] + \Pr[\mathsf{AskH3w12}_3] + \Pr[\mathsf{AskH4w123}_3] .$$

Since the authenticators are computed with the private oracles, they cannot be guessed by the adversary, better than at random for each attempt, unless the same partial transcript $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}))$ appeared in another session with real instances SG^μ and S^ν. But such a case has already been excluded in Game $\mathbf{G}_2$. Similarly, the session key cannot be distinguished by the adversary better than $1/2$:

$$\Pr[\mathsf{S}_3] \le \frac{q_{\mathsf{sendSG}}}{2^{l_1+l_2}} + \frac{q_{\mathsf{sendS}}}{2^{l_3}} + \frac{1}{2} . \qquad (\mathrm{A}\cdot 4)$$

When collisions of the partial transcripts have been excluded, the event $\mathsf{AskH12}$ can be split into three disjoint sub-cases:

- $\mathsf{AskH12\text{-}Passive}_3$: the transcript $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}))$ comes from an execution between instances of SG and S ($\mathsf{Execute}$-queries or forward of Send-queries, relay of part of them). This means that both $(t, \{X_i^*\})$ and $(Y, \{Z_j\})$ have been simulated;
- $\mathsf{AskH12\text{-}WithSG}_3$: the execution involved an instance of SG, but $(Y, \{Z_j\})$ has not been sent by any instance of S. This means that $(t, \{X_i^*\})$ has been simulated†, but $(Y, \{Z_j\})$ has been produced by the adversary;
- $\mathsf{AskH12\text{-}WithS}_3$: the execution involved an instance of S, but $(t, \{X_i^*\})$ has not been sent by any instance of SG. This means that $(Y, \{Z_j\})$ has been simulated, but $(t, \{X_i^*\})$ has been produced by the adversary.

Game $\mathbf{G}_4$: In order to evaluate the above events, we show how to embed a random Diffie-Hellman instance (P, Q), where both P and Q are generators of $\mathbb{G}_p$, and simulate with that. First, we introduce the element Q as follows:

► **Rule Setup**$^{(4)}$
Set $h \leftarrow Q$.

Next, we embed P of the Diffie-Hellman instance in the simulation of the party S.

► **Rule S1**$^{(4)}$
Choose a random element $y \overset{R}{\leftarrow} \mathbb{Z}_p^\star$ and compute $Y \equiv P^y$.

Note that we excluded the case $Y \equiv 1$. From the above simulation, we can easily see that

$$\Pr[\mathsf{S}_4] \approx \Pr[\mathsf{S}_3] . \qquad (\mathrm{A}\cdot 5)$$

Game $\mathbf{G}_5$: In this game, we first bound the probability of the event AskH_5 (or, the sub-cases). At the end of this game, we analyze on-line dictionary attacks by simply using cardinalities of some sets because the password is never used during the simulation.

Let us consider a communication between an instance S^ν and either the adversary or an instance SG^μ. For some pairs $((t, \{X_i^*\}), (Y, \{Z_j\}))$, there are two events (denoted by Guess_5 and CollW_5) to be explained below.

$$|\Pr[\mathsf{AskH}_5] - \Pr[\mathsf{AskH}_4]| \le \Pr[\mathsf{Guess}_5] + \Pr[\mathsf{CollW}_5] .$$

The Guess_5 is an event to correctly guess the secret $g^{\mathbf{s}}$

†At least one X_i^* for uncorrupted C_i has been simulated.

from V_S, and it is clearly bounded by:

$$\Pr[\mathsf{Guess}_5] \leq \frac{q_{\mathsf{hashH2}}}{|\mathbb{G}_p|} . \quad (\mathrm{A}\cdot 6)$$

Now, we consider the event CollW_5 that there are two distinct elements W_{j0} and W_{j1}, such that the tuple $(C_j, X_j^*, Y, Z_j, \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_j, Y), W_j)$ is in $\Lambda_{\mathcal{H}}$. The event CollW_5 can be upper-bounded by the following lemma:

Lemma 1: If for any pair $(X_j^*, Y) \in \left(\mathbb{G}_p\right)^2$, involved in a communication with an instance S^ν, there are two elements W_{j0} and W_{j1} such that the tuple $(C_j, X_j^*, Y, Z_j, K_{jm} \stackrel{\mathsf{def}}{=} \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_{jm}, Y), W_{jm})$ is in $\Lambda_{\mathcal{H}}$, one can solve the computational Diffie-Hellman problem:

$$\Pr[\mathsf{CollW}_5] \leq q_{\mathsf{hashH1}}^2 \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + \tau_e) . \quad (\mathrm{A}\cdot 7)$$

Proof. We prove this lemma by showing the reduction to the CDH problem when event CollW_5 happens. We assume that there exist $(X_j^*, Y \equiv P^y) \in \left(\mathbb{G}_p\right)^2$ involved in a communication with an instance S^ν, and two elements $W_{j0} \equiv Q^{\mathsf{w}_{j0}}$ and $W_{j1} \equiv Q^{\mathsf{w}_{j1}}$ such that the tuple $(C_j, X_j^*, Y, Z_j, K_{jm} \stackrel{\mathsf{def}}{=} \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_{jm}, Y), W_{jm})$ is in $\Lambda_{\mathcal{H}}$, for $m = 0, 1$. Then,

$$\begin{aligned} K_{jm} &= \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_{jm}, Y) \\ &= \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^* \times Q^{-\mathsf{w}_{jm}}, Y) \\ &= \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*, Y) \times \mathsf{CDH}_{g,\mathbb{G}_p}(Q, Y)^{-\mathsf{w}_{jm}} \\ &= \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*, Y) \times \mathsf{CDH}_{g,\mathbb{G}_p}(P, Q)^{y(-\mathsf{w}_{jm})} . \end{aligned}$$

As a consequence,

$$K_{j1}/K_{j0} = \mathsf{CDH}_{g,\mathbb{G}_p}(P, Q)^{y(\mathsf{w}_{j0} - \mathsf{w}_{j1})}$$

and thus $\mathsf{CDH}_{g,\mathbb{G}_p}(P, Q) = (K_{j1}/K_{j0})^\psi$ where ψ is the inverse of $y(\mathsf{w}_{j0} - \mathsf{w}_{j1})$ in $\mathbb{Z}_p^\star$. The latter exists since $W_{j0} \neq W_{j1}$ and $y \neq 0$. By guessing the two queries asked to the $\mathcal{H}_1$, one concludes the proof. □

In order to complete the proof, we separately bound the three sub-cases of $\mathsf{AskH12}_5$, $\mathsf{AskH3w12}_5$ and $\mathsf{AskH4w123}_5$.

- $\mathsf{AskH12\text{-}Passive}_5$: About the passive transcripts (in which both $(t, \{X_i^*\})$ and $(Y, \{Z_j\})$ have been simulated), one can state the following lemma:

  Lemma 2: If for any pair $(X_j^*, Y) \in \left(\mathbb{G}_p\right)^2$, involved in a passive transcript, there is an element W_j such that $(C_j, X_j^*, Y, Z_j, K_j \stackrel{\mathsf{def}}{=} \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_j, Y), W_j)$ is in $\Lambda_{\mathcal{H}}$, one can solve the computational Diffie-Hellman problem:

  $$\begin{aligned} &\Pr[\mathsf{AskH12\text{-}Passive}_5] \\ &\quad \leq q_{\mathsf{hashH1}} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + 2\tau_e) . \end{aligned} \quad (\mathrm{A}\cdot 8)$$

  Proof. We prove this lemma by showing the reduction to the CDH problem when event $\mathsf{AskH12\text{-}Passive}_5$ happens. We assume that there exist $(X_j^* \equiv g^{x_j}, Y \equiv P^y) \in \left(\mathbb{G}_p\right)^2$ involved in a passive transcript and $W_j \equiv Q^{\mathsf{w}_j}$ such that the tuple $(C_j, X_j^*, Y, Z_j, K_j \stackrel{\mathsf{def}}{=} \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_j, Y), W_j)$ is in $\Lambda_{\mathcal{H}}$. As above,

  $$\begin{aligned} K_j &= \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*, Y) \times \mathsf{CDH}_{g,\mathbb{G}_p}(Q, Y)^{-\mathsf{w}_j} \\ &= P^{x_j y} \times \mathsf{CDH}_{g,\mathbb{G}_p}(P, Q)^{-y\mathsf{w}_j} . \end{aligned}$$

  As a consequence, $\mathsf{CDH}_{g,\mathbb{G}_p}(P, Q) = (K_j/P^{x_j y})^\psi$ where ψ is the inverse of $-y\mathsf{w}_j$ in $\mathbb{Z}_p^\star$. The latter exists since we have excluded the cases where $y = 0$ and $\mathsf{w}_j = 0$. By guessing the query asked to the $\mathcal{H}_1$, one can get the above result. □

- $\mathsf{AskH12\text{-}WithSG}$: This corresponds to an attack where the adversary tries to impersonate S to SG. But, each authenticator V_j (sent by the adversary) for client C_j has been determined from only one corresponding $\mathsf{w}_j \leftarrow \mathcal{G}(j, pw_j)$. Therefore, the maximal probability for the adversary (who controls up to $(t-1)$ clients by invoking the $\mathsf{Register}$-queries) over a random password can be obtained by

  $$\Pr[\mathsf{AskH12\text{-}WithSG}_5] \leq \frac{q_{\mathsf{sendSG}}}{N} . \quad (\mathrm{A}\cdot 9)$$

- $\mathsf{AskH12\text{-}WithS}$: The above Lemma 1, applied to games where the event CollW_5 did not happen, states that for a pair (X_j^*, Y), involved in a transcript with an instance S^ν, there is at most one element $W_{i=j}$, such that for $W_i \equiv h^{\mathsf{w}_i}$ and $\mathsf{w}_i \leftarrow \mathcal{G}(i, pw_i)$ the corresponding tuple is in $\Lambda_{\mathcal{H}}$. When $t = n$, the maximal probability for the adversary over random passwords can be obtained by

  $$\Pr[\mathsf{AskH12\text{-}WithS}_5] \leq \frac{nq_{\mathsf{sendS}}}{N} . \quad (\mathrm{A}\cdot 10)$$

About $\mathsf{AskH3w12}$ and $\mathsf{AskH4w123}$, it means that only executions with an instance of S (and either SG or the adversary) may lead to acceptance. We can apply the same analysis as for $\mathsf{AskH12\text{-}Passive}$ and $\mathsf{AskH12\text{-}WithS}$ with the exception that the random space is $\mathbb{G}_p$ instead of N. This leads to

$$\begin{aligned} &\Pr[\mathsf{AskH3w12}_5] + \Pr[\mathsf{AskH4w123}_5] \\ &\quad \leq 2q_{\mathsf{hashG2}} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + 2\tau_e) \\ &\qquad + \frac{2nq_{\mathsf{sendS}}}{|\mathbb{G}_p|} . \end{aligned} \quad (\mathrm{A}\cdot 11)$$

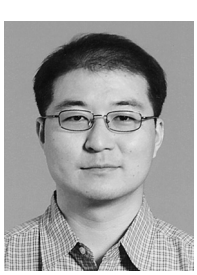
As a conclusion, we get an upper-bound for the probability of AskH_5 by combining all the cases:

$$\begin{aligned} &\Pr[\mathsf{AskH}_5] \\ &\quad \leq 3q_{\mathsf{hashG2}} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + 2\tau_e) \\ &\qquad + \frac{q_{\mathsf{sendSG}} + nq_{\mathsf{sendS}}}{N} + \frac{2nq_{\mathsf{sendS}}}{|\mathbb{G}_p|} . \end{aligned} \quad (\mathrm{A}\cdot 12)$$

Note that $q_{\mathsf{hashH1}} \leq q_{\mathsf{hashG2}}$ since a $\mathcal{H}_1$-query can invoke a $\mathcal{G}_2$-query with the partial query input.
By combining Eq. (A· 2), (A· 3), (A· 4), (A· 5), (A· 6), (A· 7) and (A· 12), one gets

$$\begin{aligned}
\Pr[\mathsf{S}_0] \leq\ & \frac{q_{\mathsf{sendSG}} + nq_{\mathsf{sendS}}}{N} \\
& + 3q_{\mathsf{hashG2}} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + 2\tau_e) \\
& + q^2_{\mathsf{hashH1}} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + \tau_e) \\
& + \frac{2nq_{\mathsf{sendS}}}{|\mathbb{G}_p|} + \frac{q_{\mathsf{hashH2}}}{|\mathbb{G}_p|} \\
& + \frac{q_{\mathsf{sendSG}}}{2^{l_1+l_2}} + \frac{q_{\mathsf{sendS}}}{2^{l_3}} + \frac{1}{2} \\
& + \frac{(q_{\mathsf{execute}} + q_{\mathsf{send}})^2}{2|\mathbb{G}_p|} \times \left(1 + \frac{1}{|\mathbb{G}_p|}\right) \\
& + \frac{q^2_{\mathsf{hashG}} + q^2_{\mathsf{hashG2}}}{2|\mathbb{G}_p|} + \frac{q^2_{\mathsf{hashH}}}{2^{l+1}} \\
\leq\ & \frac{q_{\mathsf{sendSG}} + nq_{\mathsf{sendS}}}{N} + \frac{1}{2} \\
& + 3q^2_{\mathsf{hashG2}} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + 3\tau_e) \\
& + \frac{2nq_{\mathsf{sendS}} + q_{\mathsf{hashH2}}}{|\mathbb{G}_p|} \\
& + \frac{(q_{\mathsf{execute}} + q_{\mathsf{send}} + q_{\mathsf{hashG}} + q_{\mathsf{hashG2}})^2}{|\mathbb{G}_p|} \\
& + \frac{q_{\mathsf{sendSG}}}{2^{l_1+l_2}} + \frac{q_{\mathsf{sendS}}}{2^{l_3}} + \frac{q^2_{\mathsf{hashH}}}{2^{l+1}} . \qquad \text{(A· 13)}
\end{aligned}$$

Finally, the security result as desired is obtained by noting Eq. (A· 1). □



SeongHan Shin received the B.S. and M.S. degrees in computer science from Pukyong National University, Busan, Korea, in 2000 and 2002, respectively. In 2005, he received his Ph.D. degree in information and communication engineering, information science and technology from the University of Tokyo, Tokyo, Japan. From October 2005 to March 2006, he was a post-doctoral researcher in the Institute of Industrial Science, the University of Tokyo. From April 2007 to March 2008, he was a visiting researcher in the Institute of Science and Engineering, Chuo University, Tokyo, Japan. From April 2006, he has been working for the Research Center for Information Security, National Institute of Industrial Science and Technology, Japan, as a research scientist of the Research Team for Security Fundamentals. From October 2008, he also has been serving as an associate professor in the Center for Research and Development Initiative, Chuo University, Japan. He received the CSS Student Paper Award and the IWS2005/WPMC'05 Best Student Paper Award in 2003 and 2005, respectively. His research interests include information security, cryptography and wireless security.

Kazukuni Kobara received his B.E. degree in electrical engineering and M.E. degree in computer science and system engineering from the Yamaguchi University in 1992, 1994, respectively. He also received his Ph.D. degree in engineering from the University of Tokyo in 2003. From 1994 to 2000 and 2000 to 2006 he was a technical associate and a research associate respectively for the Institute of Industrial Science of the University of Tokyo. In 2006, he joined the National Institute of Advanced Industrial Science and Technology (AIST) where he was the leader of the Research Team for Security Fundamentals in the Research Center for Information Security (RCIS). Currently he is a chief research scientist at RCIS. His research interests include cryptography, information and network security. He received the SCIS Paper Award and the Vigentennial Award from ISEC group of IEICE in 1996 and 2003, respectively. He also received the Best Paper Award of WISA, the ISITA Paper Award for Young Researchers, the IEICE Best Paper Award (Inose Award), the WPMC Best Paper Award and the JSSM Best Paper Award in 2001, 2002, 2003, 2005 and 2006 respectively. He is a member of IEICE and IACR. He served as a member of CRYPTREC (2000-present), the vice chairperson of WLAN security committee (2003) and the chief investigator of INSTAC identity management committer (2007-present).

Hideki Imai was born in Shimane, Japan on May 31, 1943. He received the B.E., M.E., and Ph.D. degrees in electrical engineering from the University of Tokyo in 1966, 1968, and 1971, respectively. From 1971 to 1992 he was on the faculty of Yokohama National University. From 1992 to 2006 he was a Professor in the Institute of Industrial Science, the University of Tokyo. In 2006 he was appointed as an Emeritus Professor of the University of Tokyo and a Professor of Chuo University. Concurrently he serves as the Director of Research Center for Information Security, National Institute of Advanced Industrial Science and Technology. His current research interests include information theory, coding theory, cryptography, and information security. From IEICE (the Institute of Electronics, Information and Communication Engineers), Dr. Imai received Best Book Awards in 1976 and 1991, Best Paper Awards in 1992, 2003 and 2004, Yonezawa Memorial Paper Award in 1992, Achievement Award in 1995, Inose Award in 2003, and Distinguished Achievement and Contributions Award in 2004. He also received Golden Jubilee Paper Award from the IEEE Information Theory Society in 1998, Wilkes Award from the British Computer Society in 2007, Official Commendations from the Minster of Internal Affairs and Communications in June 2002 and from the Minister of Economy, Trade and Industry in October 2002. He was awarded Honor Doctor Degree by Soonchunhyang University, Korea in 1999 and Docteur Honoris Causa by the University of Toulon Var, France in 2002. He is also the recipient of the Ericsson Telecommunications Award 2005 and the Okawa Prize 2008. Dr. Imai is a member of the Science Council of Japan. He was elected a Fellow of IEEE, IEICE and IACR (International Association for Cryptologic Research) in 1992, 2001 and 2007, respectively. He is an IEEE life Fellow since 2009. He has chaired many committees of scientific societies and organized a number of international conferences. He served as the President of the Society of Information Theory and its Applications in 1997, of the IEICE Engineering Sciences Society in 1998, and of the IEEE Information Theory Society in 2004. He is currently the Chair of CRYPTREC (Cryptography Techniques Research and Evaluation Committee of Japan).