A Secure Threshold Anonymous Password-Authenticated Key Exchange Protocol [⋆]

SeongHan Shin[1], Kazukuni Kobara[1], and Hideki Imai[2,1]

[1] Research Center for Information Security (RCIS),
National Institute of Advanced Industrial Science and Technology (AIST),
1-18-13, Sotokanda, Chiyoda-ku, Tokyo, 101-0021 Japan
E-mail: `seonghan.shin@aist.go.jp`
[2] Chuo University,
1-13-27, Kasuga, Bunkyo-ku, Tokyo, 112-8551 Japan

(submitted on December 22nd, 2008)

Abstract. At Indocrypt 2005, Viet et al., [22] 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 rigorously prove that the TAP protocol has semantic security of session keys in the random oracle model by showing the reduction to the computational Diffie-Hellman problem. In addition, the TAP protocol provides unconditional 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 [22].

Key words: password authentication, key exchange, PAKE, anonymity, provable security

Preface

At Indocrypt 2008, Yang and Zhang [25] have shown two attacks on the TAP (threshold $t \geq 2$) protocol, and then proposed the NAPAKE (i.e., $t = 1$) and D-NAPAKE (i.e., $t \geq 2$) protocols. Here, we add some comments on their paper [25].

About two attacks on the TAP ($t \geq 2$) protocol. In [25], they showed two *insider attacks* on legitimate clients in the TAP ($t \geq 2$) protocol. However, we proved AKE security and unilateral authentication of the TAP ($t \geq 2$) protocol against an adversary $\mathcal{A} \notin \{C_1, \cdots, C_n, S\}$ where $C = \{C_1, \cdots, C_n\}$ is a set of all clients and S is the server (see the security model in Section 4). Of course, we agree that considering insider attacks and finding a solution are one of the research directions in cryptography. In Appendix B, we give a simple countermeasure for the TAP ($t \geq 2$) protocol against the two attacks (i.e., impersonation attack and off-line dictionary attack). In fact, we considered keyword search as an application of the TAP ($t \geq 2$) protocol and, in such applications, the off-line dictionary attack of legitimate clients is not possible because each share doesn't need to be transmitted to other parties.

The D-NAPAKE protocol is not threshold anonymous PAKE! In Appendix C, we show an attack on the D-NAPAKE (i.e., $t \geq 2$) protocol of [25] where only one legitimate client can impersonate any subgroup of clients to the server. That actually means that the D-NAPAKE ($t \geq 2$) protocol is **NOT** a threshold anonymous PAKE protocol unlike the author's claim.

[⋆] This is the full version of [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, 17]), 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, 24, 2, 17, 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 (PAKE) and Anonymity

In 1992, Bellovin and Merritt [4] discussed an interesting problem about how to design a secure password-only protocol where a client remembers his/her password only 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., [22] have proposed an anonymous PAKE protocol and its threshold construction[3] that simply combine a PAKE protocol [1] for generating secure channels with an Oblivious Transfer (OT) protocol [21, 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

---

[3] In their construction, the "threshold" number of clients collaborate one another to make a subgroup of the whole clients' group. In a different context, MacKenzie et al., [14] proposed a threshold PAKE protocol where the "threshold" number of servers collaborate 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).

can upload opinions in a password-authenticated and anonymous way. As discussed in [22], 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 [22]). 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 [15] on the relationship between PAKE protocols and other cryptographic primitives, we carefully revisit Viet et al's anonymous PAKE protocols [22]. 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 deduce 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 [22].

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. In Section 3, we propose a secure threshold anonymous PAKE (TAP) protocol. Section 4 and 5 are devoted to its security model and 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.

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 [22, 23] 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 $d \overset{R}{\leftarrow} D$ indicates the process of selecting d at random and uniformly over D. Let "$||$" denote the concatenation of bit strings in $\{0,1\}^\star$. Let "$\bigoplus$" denote the exclusive-OR (XOR) operation of bit strings. The hash functions $\mathcal{F}$ and $\mathcal{F}'$ are full-domain hash (FDH) functions, mapping $\{0,1\}^\star$ to $\mathbb{Z}_p^\star$. While $\mathcal{G} : \{0,1\}^\star \rightarrow \mathbb{G}_p$ is another FDH function, the others are denoted $\mathcal{H}_k : \{0,1\}^\star \rightarrow \{0,1\}^l$, 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 $\in \{0,1\}^\star$.

Public information: $(\mathbb{G}_p, p, g, h), \mathcal{F}, \mathcal{F}', \mathcal{G}, \mathcal{H}_1, \mathcal{H}_2, C = \{C_1, \cdots, C_n\}$

Subgroup $SG = \{C_1, \cdots, C_t\}$

$x \overset{R}{\leftarrow} \mathbb{Z}_p^\star, X \equiv g^x$

For each C_i $(1 \le i \le t)$,

$(r_i, s_i) \overset{R}{\leftarrow} \left(\mathbb{Z}_p^\star\right)^2$,

$w_i \equiv h^{r_i \cdot \mathcal{F}(i, pw_{C_i})}$,

and $A_i \equiv w_i \times g^{s_i}$.

For $l = t+1$ to n, $w_l \overset{R}{\leftarrow} \mathbb{G}_p$.

$$\xrightarrow{C, X, \{A_i\}_{1 \le i \le t}, \{w_j\}_{1 \le j \le n}}$$

Server S $\left(pw_{C_j},\ 1 \le j \le n\right)$

$(y, z) \overset{R}{\leftarrow} \left(\mathbb{Z}_p^\star\right)^2, Y \equiv g^y, Z \equiv g^z$

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

$u_0 = y$ and $u_k \overset{R}{\leftarrow} \mathbb{Z}_p^\star$ $(1 \le k \le t-1)$

For $j = 1$ to n,

$y_j \leftarrow f(j)$,

$Y_j \equiv g^{y_j}$,

and $\alpha_j \leftarrow \mathcal{G}\left(w_j^z\right) \bigoplus \left(Y_j \times g^{\mathcal{F}'(pw_{C_j})}\right)$.

For $i = 1$ to t, $D_i \equiv A_i^z$.

$K_S \equiv X^y$

$V_S \leftarrow \mathcal{H}_1(T || \{D_i\}_{1 \le i \le t} || \{\alpha_j\}_{1 \le j \le n} || Y || K_S)$

$SK \leftarrow \mathcal{H}_2(T || \{D_i\}_{1 \le i \le t} || \{\alpha_j\}_{1 \le j \le n} || Y || K_S)$

$$\xleftarrow{S, Z, \{D_i\}_{1 \le i \le t}, \{\alpha_j\}_{1 \le j \le n}, V_S}$$

For each C_i $(1 \le i \le t)$,

$Y_i \equiv (\alpha_i \bigoplus \mathcal{G}(D_i / Z^{s_i})) \times \left(g^{\mathcal{F}'(pw_{C_i})}\right)^{-1}$.

$Y \equiv \prod_{i=1}^{t} Y_i^{\lambda_i}$ where $\lambda_i \equiv \prod_{1 \le k \le t, k \ne i} \frac{k}{k-i}$,

and $K_C \equiv Y^x$

If $V_S \ne \mathcal{H}_1(T || \{D_i\}_{1 \le i \le t} || \{\alpha_j\}_{1 \le j \le n} || Y || K_C)$, reject.

Otherwise, $SK \leftarrow \mathcal{H}_2(T || \{D_i\}_{1 \le i \le t} || \{\alpha_j\}_{1 \le j \le n} || Y || K_C)$

and accept.

Fig. 1. The threshold anonymous PAKE (TA-PAKE) protocol [22, 23] where $T = C || S || X || Z$

2.2 Protocol Description

Here, we describe the threshold anonymous PAKE (TA-PAKE) protocol [22, 23] where any subgroup SG, consisting of at least t $(t \le n)$ clients among n clients, generates a session key with server S in a password-authenticated and anonymous way.[4] We assume that each client in the subgroup are connected via secure channels. See Fig. 1 for a 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 $\mathbb{Z}_p^\star$ and computes $X \equiv g^x$.

1.2 Each client C_i $(1 \le i \le t)$ chooses two random numbers $(r_i, s_i) \overset{R}{\leftarrow} \left(\mathbb{Z}_p^\star\right)^2$, 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.

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

Step 2

2.1 The server chooses two random numbers $(y, z) \overset{R}{\leftarrow} \left(\mathbb{Z}_p^\star\right)^2$ 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 [16]. Specifically, server S generates the respective share $f(j)$, for n clients, from a random polynomial $f(x)$ of

---

[4] The only difference of [23] from [22] is that the subgroup SG chooses $w_l \overset{R}{\leftarrow} \mathbb{G}_p$, for $t + 1 \le l \le n$, and sends $\{w_j\}_{1 \le j \le n}$ along with other values in the first flow. In fact, the TA-PAKE protocol of [22] doesn't work correctly since server S has no idea on w_j and cannot compute α_j that needs w_j in the computation.

degree $t-1$ with coefficients u_k $(1 \leq k \leq t-1)$ in $\mathbb{Z}_p^\star$

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

and sets $u_0 = y$.

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

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 $Y_i \equiv (\alpha_i \bigoplus \mathcal{G}(D_i/Z^{s_i})) \times \left(g^{\mathcal{F}'(pw_{C_i})}\right)^{-1}$.

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 [22, 23] 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 $\mathbb{Z}_p^\star$ and computes $X \equiv g^x$, and also chooses n random numbers $w_j \overset{R}{\leftarrow} \mathbb{G}_p$, 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 $s_i \overset{R}{\leftarrow} \mathbb{Z}_p^\star$ 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 $(1 \leq i \leq t)$.

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

.

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 $\{pw'_{C_1}, \cdots, pw'_{C_t}\}$ 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 $\{pw'_{C_1}, \cdots, pw'_{C_t}\}$ 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. Also, keep in mind that the threshold t is controlled by the adversary.

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.

Public information: $(\mathbb{G}_p, p, g, h), \mathcal{F}, \mathcal{G}, \mathcal{H}_1, \mathcal{H}_2, C = \{C_1, \cdots, C_n\}$

Subgroup $SG = \{\underbrace{C_1, C_4, \cdots, C_{n-3}}_{t}\}$
For each $C_i \in SG$,
$x_i \xleftarrow{R} \mathbb{Z}_p^\star$, $X_i \equiv g^{x_i}$,
$W_i \equiv h^{\mathcal{F}(i, pw_{C_i})}$,
and $X_i^* \equiv X_i \times W_i$.
For each $C_j \in C \backslash SG$ $(1 \leq j \neq i \leq n)$,
$X_j^* \xleftarrow{R} \mathbb{G}_p$.

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

Server S $\left(W_j \equiv h^{\mathcal{F}(j, pw_{C_j})},\ 1 \leq j \leq n\right)$
$y \xleftarrow{R} \mathbb{Z}_p^\star, Y \equiv g^y, \mathsf{S} \xleftarrow{R} \mathbb{G}_p$
$f(x) \equiv \sum_{k=0}^{t-1} u_k \cdot x^k$ where
$u_0 = \mathsf{S}$ and $u_k \xleftarrow{R} \mathbb{G}_p$ $(1 \leq k \leq t-1)$
For $j = 1$ to n,
$\mathsf{S}_j \leftarrow f(j)$,
$X_j \equiv X_j^* / W_j$,
$K_j \equiv X_j^y$,
and $Z_j \leftarrow \mathcal{G}(j, K_j) \bigoplus \mathsf{S}_j$.
$V_S \leftarrow \mathcal{H}_1(C||S||t||\{X_i^*\}_{1 \leq i \leq n}||Y||\{Z_j\}_{1 \leq j \leq n}||\mathsf{S})$

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

$SK \leftarrow \mathcal{H}_2(C||S||t||\{X_i^*\}_{1 \leq i \leq n}||Y||\{Z_j\}_{1 \leq j \leq n}||\mathsf{S})$

For each $C_i \in SG$,
look for $Z_{j=i}$,
$K_i \equiv Y^{x_i}$ and $\mathsf{S}_i = Z_i \bigoplus \mathcal{G}(i, K_i)$.
$\mathsf{S}' = \sum_{k=1}^{t} \lambda_k \cdot \mathsf{S}_k$ where $\lambda_k \equiv \prod_{1 \leq m \leq t, m \neq k} \frac{m}{m-k}$
If $V_S \neq \mathcal{H}_1(C||S||t||\{X_i^*\}_{1 \leq i \leq n}||Y||\{Z_j\}_{1 \leq j \leq n}||\mathsf{S}')$, reject.
Otherwise, $SK \leftarrow \mathcal{H}_2(C||S||t||\{X_i^*\}_{1 \leq i \leq n}||Y||\{Z_j\}_{1 \leq j \leq n}||\mathsf{S}')$
and accept.

Fig. 2. A secure threshold anonymous PAKE (TAP) protocol where the threshold $t > 1$

3 A Secure Threshold Anonymous PAKE 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 [22]. 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 51. 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 52 in Section 5).

Step 1

1.1 Each client C_i, who belongs to the subgroup SG, chooses a random number x_i from $\mathbb{Z}_p^\star$ and computes the Diffie-Hellman public value $X_i \equiv g^{x_i}$. The client C_i also computes the password verification data $W_i \equiv h^{\mathcal{F}(i, pw_{C_i})}$ 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 chooses $X_j^* \xleftarrow{R} \mathbb{G}_p$ 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 and $\{X_i^*\}_{1 \leq i \leq n}$, to the server, together with the set C of clients' identities.

Step 2

2.1 The server S chooses a random number y from $\mathbb{Z}_p^\star$ 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 [16]. 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 $u_0 = \mathsf{S}$ 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 and the hashed output of index j and K_j: $Z_j \leftarrow \mathcal{G}(j, K_j) \bigoplus \mathsf{S}_j$ where $\mathsf{S}_j \leftarrow f(j)$.

2.3 Also server S generates an authenticator $V_S \leftarrow \mathcal{H}_1(C||S||t||\{X_i^*\}_{1 \leq i \leq n}||Y||\{Z_j\}_{1 \leq j \leq n}||\mathsf{S})$ and a session key $SK \leftarrow \mathcal{H}_2(C||S||t||\{X_i^*\}_{1 \leq i \leq n}\ ||Y||\{Z_j\}_{1 \leq j \leq n}||\mathsf{S})$. 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: $\mathsf{S}_i = Z_i \bigoplus \mathcal{G}(i, K_i)$.

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

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

Remark. In order to provide mutual authentication in the above protocol, we can simply add the subgroup's authenticator $V_{SG} \leftarrow \mathcal{H}_3(C||S||t||\{X_i^*\}_{1 \leq i \leq n}||Y||\{Z_j\}_{1 \leq j \leq n}||\mathsf{S}')$, 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].

4 The Model, Security Notions and Mathematical Assumption

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

4.1 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^μ (resp., S^ν) where $\mu, \nu \in \mathbb{N}$, or by U in case of any instance. 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^μ, S^ν): 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.

- $\mathsf{Send}(U, m)$: 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 $\mathsf{Send}(SG^{\mu}, \mathtt{Start})$ initializes the key exchange protocol, and thus the adversary receives the first flow.
- $\mathsf{Reveal}(U)$: 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 $\mathcal{A}$.
- $\mathsf{Test}(U)$: This query is used to see whether the adversary can obtain some information on the session key or not. 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 ($\mathsf{Reveal}(U)$ would output) if $b = 1$, or a random value with the same size except the session key if $b = 0$.

4.2 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 $\mathbf{Game}^{\mathsf{ake}}(\mathcal{A}, P)$, in which 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. The protocol P is said to be (t, ε)-AKE-secure if $\mathcal{A}$'s advantage is smaller than ε for any adversary $\mathcal{A}$ running time t.

Another goal is to consider unilateral authentication of either SG ($\mathsf{SG\text{-}auth}$) or S ($\mathsf{S\text{-}auth}$) wherein the adversary impersonates a party. We denote by $\mathsf{Succ}_P^{\mathsf{SG-auth}}(\mathcal{A})$ (resp., $\mathsf{Succ}_P^{\mathsf{S-auth}}(\mathcal{A})$) 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, ε)-Auth-secure if $\mathcal{A}$'s success probability for breaking either $\mathsf{SG\text{-}auth}$ or S-auth is smaller than ε for any adversary $\mathcal{A}$ running time t.

By following the definition of anonymity from [22], 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.

4.3 Computational Diffie-Hellman Assumption

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

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]. Note that secure unilateral authentication can be easily extended to mutual authentication by adding another authenticator as suggested in [5, 3].

Theorem 51 (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 $n \leq 2\sqrt{N-1}-1$.*[5] *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., $\mathcal{G}$), $\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) \leq 4\varepsilon$ and $\mathsf{Adv}_P^{\mathsf{S-auth}}(\mathcal{A}) \leq \varepsilon$, with ε upper-bounded by*

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

where l_1 is the output length of $\mathcal{H}_1$ and 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 (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. The proof can be found in Appendix A.

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

Theorem 52 *The* TAP *protocol provides client's anonymity against a passive server in an information-theoretic sense.*

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 $C_j \in C \backslash SG$) since they are completely independent one another. In addition, even if server S receives the subgroup's authenticator $V_{SG} \leftarrow \mathcal{H}_3(C||S||t||\{X_i^*\}_{1\leq i\leq n}||Y||\{Z_j\}_{1\leq j\leq n}||\mathsf{S}')$ 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. □

6 The Condition on n

Here we deduce the condition on n, appeared in Theorem 51, 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 61 *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 $\mathbf{G}_5$** in the proof, we explain why the probability of on-line dictionary attacks is upper-bounded by the above. In order to maximize $\Pr[\mathsf{AskH1\text{-}WithSG}_5]$, 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 $\Pr[\mathsf{AskH1\text{-}WithS}_5]$, the same discussion can be applied. □

---

[5] In practice, $N = 2^{37}$ for MS-Windows passwords. It is sufficiently large for n.

Public information: $(\mathbb{G}_p, p, g, h), \mathcal{F}, \mathcal{G}, \mathcal{H}_1, \mathcal{H}_2, C = \{C_1, \cdots, C_n\}$

Client C_i (pw_{C_i})		Server S $\left(W_j \equiv h^{\mathcal{F}(j,pw_{C_j})},\ 1 \leq j \leq n\right)$
$x \xleftarrow{R} \mathbb{Z}_p^\star$, $X \equiv g^x$		$y \xleftarrow{R} \mathbb{Z}_p^\star, Y \equiv g^y, \mathsf{S} \xleftarrow{R} \{0,1\}^l$
$W_i \equiv h^{\mathcal{F}(i,pw_{C_i})}$, $X^* \equiv X \times W_i$	$\xrightarrow{C, X^*}$	For $j = 1$ to n,
		$X_j \equiv X^*/W_j$,
		$K_j \equiv X_j^y$,
		and $Z_j \leftarrow \mathcal{G}(j, K_j) \bigoplus \mathsf{S}$.
	$\xleftarrow{S, Y, \{Z_j\}_{1 \leq j \leq n}, V_S}$	$V_S \leftarrow \mathcal{H}_1(C \Vert S \Vert X^* \Vert Y \Vert \{Z_j\}_{1 \leq j \leq n} \Vert \mathsf{S})$
For $i = j$, $\mathsf{S}' = Z_i \bigoplus \mathcal{G}(i, Y^x)$.		
If $V_S \neq \mathcal{H}_1(C \Vert S \Vert X^* \Vert Y \Vert \{Z_j\}_{1 \leq j \leq n} \Vert \mathsf{S}')$, reject.		$SK \leftarrow \mathcal{H}_2(C \Vert S \Vert X^* \Vert Y \Vert \{Z_j\}_{1 \leq j \leq n} \Vert \mathsf{S})$
Otherwise, $SK \leftarrow \mathcal{H}_2(C \Vert S \Vert X^* \Vert Y \Vert \{Z_j\}_{1 \leq j \leq n} \Vert \mathsf{S}')$		
and accept.		

Fig. 3. An efficient anonymous PAKE protocol when $t = 1$

Table 1. Comparison of anonymous PAKE protocols as for efficiency where n is the number of clients

Protocols	The number of modular exponentiations		Communication bandwidth
	Client C_i	Server S	
APAKE [22]	6 (4)	$4n + 2$ $(3n + 1)$	$\lvert C\rvert + \lvert S\rvert + (n+1)\lvert\mathsf{hash}\rvert + (n+2)\lvert p\rvert$
Our protocol of Fig. 3	3 (2)	$n + 1$ (n)	$\lvert C\rvert + \lvert S\rvert + (n+1)\lvert\mathsf{hash}\rvert + 2\lvert p\rvert$

Now the condition on n can be easily obtained by restricting the probability of Theorem 61 to $1/N$:

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

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 $\{0,1\}^l$. By following the security proof of Appendix A, we can remove the condition on n because the on-line attacks at **Game $\mathbf{G}_5$** is limited to one specific 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 [22]) in terms of computation costs and communication bandwidth to be required (see Table 1). 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 modular exponentiations for both client and server. With respect to communication bandwidth, our protocol requires a bandwidth of

$((n+1)|\mathsf{hash}| + 2|p|)$-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 $|\mathsf{hash}| = 160$). The gap of communication bandwidths between our and APAKE protocols becomes bigger 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 [22].

References

1. M. Abdalla and D. Pointcheval. Simple Password-Based Encrypted Key Exchange Protocols. In *Proc. of CT-RSA 2005*, LNCS 3376, pp. 191-208. Springer-Verlag, 2005.
2. M. Bellare, R. Canetti, and H. Krawczyk. 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, 1998.
3. E. Bresson, O. Chevassut, and D. Pointcheval. New Security Results on Encrypted Key Exchange. In *Proc. of PKC 2004*, LNCS 2947, pp. 145-158. Springer-Verlag, 2004.
4. S. M. Bellovin and M. Merritt. Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks. In *Proc. of IEEE Symposium on Security and Privacy*, pp. 72-84, 1992.
5. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *Proc. of EUROCRYPT 2000*, LNCS 1807, pp. 139-155. Springer-Verlag, 2000.
6. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proc. of ACM CCS '93*, pp. 62-73, 1993.
7. M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In *Proc. of CRYPTO '93*, LNCS 773, pp. 232-249. Springer-Verlag, 1993.
8. C. K. Chu and W. G. Tzeng. Efficient k-Out-of-n Oblivious Transfer Schemes with Adaptive and Non-adaptive Queries. In *Proc. of PKC 2005*, LNCS 3386, pp. 172-183. Springer-Verlag, 2005.
9. W. Diffie and M. Hellman. New Directions in Cryptography. In *IEEE Transactions on Information Theory*, Vol. IT-22(6), pp. 644-654, 1976.
10. W. Diffie, P. van Oorschot, and M. Wiener. 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. H. Krawczyk. SIGMA: the 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman and its Use in the IKE Protocols. In *Proc. of CRYPTO 2003*, LNCS 2729, pp. 400-425. Springer-Verlag, 2003.
13. P. MacKenzie. On the Security of the SPEKE Password-Authenticated Key Exchange Protocol. Cryptology ePrint Archive: Report 2001/057, `http://eprint.iacr.org/2001/057`.
14. P. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold Password-Authenticated Key Exchange. In *Proc. of CRYPTO 2002*, LNCS 2442, pp. 385-400. Springer-Verlag, 2002.
15. M. H. Nguyen. The Relationship Between Password-Authenticated Key Exchange and Other Cryptographic Primitives. In *Proc. of TCC 2005*, LNCS 3378, pp. 457-475. Springer Verlag, 2005.
16. A. Shamir. How to Share a Secret. In *Proc. of Communications of the ACM*, Vol. 22(11), pp. 612-613, 1979.
17. V. Shoup. On Formal Models for Secure Key Exchange. IBM Research Report RZ 3121, 1999. Available at `http://eprint.iacr.org/1999/012`.
18. V. Shoup. OAEP Reconsidered. *Journal of Cryptology*, Vol. 15(4), pp. 223-249, September 2002.
19. V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. Cryptology ePrint Archive: Report 2004/332, `http://eprint.iacr.org/2004/332`.

20. S. H. Shin, K. Kobara, and H. Imai. A Secure Threshold Anonymous Password-Authenticated Key Exchange Protocol. In *Proc. of IWSEC 2007*, LNCS 4752, pp. 444-458. Springer-Verlag, 2007.
21. W. G. Tzeng. Efficient 1-Out-n Oblivious Transfer Schemes. In *Proc. of PKC 2002*, LNCS 2274, pp. 159-171. Springer-Verlag, 2002.
22. D. Q. Viet, A. Yamamura, and H. Tanaka. Anonymous Password-Based Authenticated Key Exchange. In *Proc. of INDOCRYPT 2005*, LNCS 3797, pp. 244-257. Springer-Verlag, 2005.
23. D. Q. Viet, A. Yamamura, and H. Tanaka. Anonymous Password-Based Authenticated Key Exchange. In *Proc. of the 2006 Symposium on Cryptography and Information Security (SCIS 2006)*, 3D3-4, January 2006.
24. S. B. Wilson, D. Johnson, and A. Menezes. Key Agreement Protocols and their Security Analysis. In *Proc. of IMA International Conference on Cryptography and Coding*, December 1997.
25. J. Yang and Z. Zhang. A New Anonymous Password-Based Authenticated Key Exchange Protocol. In *Proc. of INDOCRYPT 2008*, LNCS 5365, pp. 200-212. Springer-Verlag, 2008.

A Proof of Theorem 51

In this proof, we incrementally define a sequence of games starting at the real game $\mathbf{G}_0$ and ending up at $\mathbf{G}_5$ where the hash functions are modelled as random oracles. We use Shoup's difference lemma [18, 19] to bound the probability of each event in these games. For visual simplicity, we denote $\{X_i^*\}_{1 \le i \le n}$ and $\{Z_j\}_{1 \le j \le n}$ by $\{X_i^*\}$ and $\{Z_j\}$, respectively, in the proof.

Game $\mathbf{G}_0$ (Real protocol): This is the real protocol in the random oracle model. We consider the following two events:

- S_0 (for semantic security) which occurs if the adversary correctly guesses the bit b involved in the Test-query;
- A_0 (for S-authentication) which occurs if an instance SG^μ accepts with no partner instance S^ν with the same transcript $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}, V_S))$

$$\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) = 2\Pr[\mathsf{S}_0] - 1, \mathsf{Adv}_P^{\mathsf{S-auth}}(\mathcal{A}) = \Pr[\mathsf{A}_0] \ . \tag{3}$$

In any game $\mathbf{G}_n$ below, we study the event A_n and the restricted event $\mathsf{SwA}_n = \mathsf{S}_n \wedge \neg\mathsf{A}_n$.

Game $\mathbf{G}_1$ (Simulation for hash and other queries): In this game, we simulate the hash oracles ($\mathcal{F}$, $\mathcal{G}$, $\mathcal{H}_1$ and $\mathcal{H}_2$, but as well additional hash functions $\mathcal{H}'_k : \{0,1\}^\star \rightarrow \{0,1\}^{l_k}$, for $k = 1, 2$, which will appear in the Game $\mathbf{G}_3$) as usual by maintaining hash lists $\mathit{\Lambda}_{\mathcal{F}}$, $\mathit{\Lambda}_{\mathcal{G}}$, $\mathit{\Lambda}_{\mathcal{H}}$ and $\mathit{\Lambda}_{\mathcal{H}'}$ (see below). Note that we do not use the private oracles for $\mathcal{F}$ and $\mathcal{G}$. We also simulate all the instances, as the real parties would do, for the Send-queries and for the $\mathsf{Execute}$, Reveal and Test-queries (see below). From this simulation, we can easily see that the game is perfectly indistinguishable from the real attack.

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

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

Simulation of the TAP protocol

Send-queries to SG

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

- A $\mathsf{Send}(SG^\mu, \mathtt{Start})$-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**$^{(1)}$

    Choose a random element $b \xleftarrow{R} \mathbb{Z}_p^\star$, and compute $h \equiv g^b$ and $W_i \equiv h^{\omega_i}$ where

$\omega_i \leftarrow \mathcal{F}(i, pw_{C_i})$.

▶ **Rule SG2**$^{(1)}$

For $C_i \in SG$, choose a random element $\theta_i \overset{R}{\leftarrow} \mathbb{Z}_q^\star$, and compute $X_i \equiv g^{\theta_i}$ and $X_i^* \equiv X_i \times W_i$. For $C_j \in C \backslash SG$, choose a random element $X_j^* \overset{R}{\leftarrow} \mathbb{G}_p$.

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 $\mathsf{Send}(SG^\mu, (S, Y, \{Z_j\}, V_S))$ is processed by reconstructing the secret S and by computing the alleged 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 $\mathsf{S}_i = Z_i \bigoplus \mathcal{G}(i, K_i)$.

▶ **Rule SG4**$^{(1)}$

Compute the expected authenticator and the session key:
$V_S' \leftarrow \mathcal{H}_1(C||S||t||\{X_i^*\}||Y||\{Z_j\}||\mathsf{S}')$, $SK_{SG} \leftarrow \mathcal{H}_2(C||S||t||\{X_i^*\}||Y||\{Z_j\}||\mathsf{S}')$.

If $V_S' = V_S$, then the instance accepts. In any case, it terminates.

Send-queries to S

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 random elements $\varphi \overset{R}{\leftarrow} \mathbb{Z}_p^\star$ and $\mathsf{S} \overset{R}{\leftarrow} \mathbb{G}_p$, and compute $Y \equiv g^\varphi$.

Then, the instance computes the authenticator and the session key after generating the shares S_j of S by (t, n)-threshold secret sharing scheme [16]. We apply the following rules:

▶ **Rule S2**$^{(1)}$

For j $(1 \le j \le n)$, compute $X_j \equiv X_j^*/W_j$, $K_j \equiv X_j^\varphi$ and $Z_j = \mathcal{G}(j, K_j) \bigoplus \mathsf{S}_j$.

▶ **Rule S3**$^{(1)}$

Compute the authenticator and the session key:
$V_S \leftarrow \mathcal{H}_1(C||S||t||\{X_i^*\}||Y||\{Z_j\}||\mathsf{S})$, $SK_S \leftarrow \mathcal{H}_2(C||S||t||\{X_i^*\}||Y||\{Z_j\}||\mathsf{S})$.

Finally, the query is answered with $(S, Y, \{Z_j\}, V_S)$, and then the instance accepts and terminates.

Other queries

- An $\mathsf{Execute}(SG^\mu, S^\nu)$-query is processed using successively the above simulations of the Send-queries: $(C, t, \{X_i^*\}) \leftarrow \mathsf{Send}(SG^\mu, \mathtt{Start})$, $(S, Y, \{Z_j\}, V_S) \leftarrow \mathsf{Send}(S^\nu, (C, t, \{X_i^*\}))$, and then outputting the transcript $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}, V_S))$.
- A $\mathsf{Reveal}(U)$-query returns the session key (SK_{SG} or SK_S) computed by the instance U (if the latter has accepted).
- A $\mathsf{Test}(U)$-query first gets SK from $\mathsf{Reveal}(U)$, 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 $\{0, 1\}^{l_2}$.

Game $\mathbf{G}_2$ (Collisions): For an easier analysis in the following, we cancel games in which some collisions (Coll_2) are unlikely to happen:

- collisions on the partial transcripts $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}))$: any 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.
- collision on the output of $\mathcal{F}$.
- collision on the output of $\mathcal{G}$.

These probabilities are upper-bounded by the birthday paradox (and when $t = 2$):

$$\Pr[\mathsf{Coll}_2] \le \frac{(q_e + q_s)^2}{2|\mathbb{G}_p|^2} \times \left(1 + \frac{1}{|\mathbb{G}_p|}\right) + \frac{q_f^2}{2p} + \frac{q_g^2}{2|\mathbb{G}_p|}\ . \tag{4}$$

Game $\mathbf{G}_3$ (Using private oracles): In order to make the authenticator and the session key unpredictable to any adversary, we compute them using the private oracles $\mathcal{H}_1'$ and $\mathcal{H}_2'$ (instead of $\mathcal{H}_1$ and

$\mathcal{H}_2$), respectively, so that the values are completely independent from the random oracles. We reach this aim by using the following rule:

▶ **Rule SG4/S3$^{(3)}$**

Compute the authenticator $V_S \leftarrow \mathcal{H}_1'(C||S||t||\{X_i^*\}||Y||\{Z_j\})$.

Compute the session key $SK_{SG/S} \leftarrow \mathcal{H}_2'(C||S||t||\{X_i^*\}||Y||\{Z_j\})$.

Since we do no longer need to compute the value S, we can simplify the following rules:

▶ **Rule SG3$^{(3)}$**

Do nothing.

▶ **Rule S1$^{(3)}$**

Choose random elements $\varphi \overset{R}{\leftarrow} \mathbb{Z}_p^\star$ and $\mathsf{T} \overset{R}{\leftarrow} \mathbb{G}_p$, and compute $Y \equiv g^\varphi$.

▶ **Rule S2$^{(3)}$**

For j $(1 \le j \le n)$, set $Z_j \leftarrow \mathsf{T}_j$ where T_j is a share of T.

Finally, the secret ω_i is not used anymore either so that we can also simplify the generation of $\{X_i^*\}$ using the group property of $\mathbb{G}_p$.

▶ **Rule SG2$^{(3)}$**

For i $(1 \le i \le n)$, choose a random element $x_i \overset{R}{\leftarrow} \mathbb{Z}_p^\star$ and compute $X_i^* \equiv g^{x_i}$.

The games $\mathbf{G}_3$ and $\mathbf{G}_2$ are indistinguishable unless some specific hash queries are asked, denoted by event $\mathsf{AskH}_3 = \mathsf{AskH1}_3 \vee \mathsf{AskH2w1}_3$:

- $\mathsf{AskH1}_3$: $\mathcal{H}_1(C||S||t||\{X_i^*\}||Y||\{Z_j\}||\mathsf{S})$ has been queried by $\mathcal{A}$ to $\mathcal{H}_1$ for some execution transcripts $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}))$;
- $\mathsf{AskH2w1}_3$: $\mathcal{H}_2(C||S||t||\{X_i^*\}||Y||\{Z_j\}||\mathsf{S})$ has been queried by $\mathcal{A}$ to $\mathcal{H}_2$ for some execution transcripts $((C, t, \{X_i^*\}), (S, Y, \{Z_j\}))$, where some party has accepted, but event $\mathsf{AskH1}_3$ did not happen;

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

$$\Pr[\mathsf{AskH}_3] \le \Pr[\mathsf{AskH1}_3] + \Pr[\mathsf{AskH2w1}_3] \ .$$

The authenticator is computed with a random oracle that is private to the simulator, then one can remark that it 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 a real instance S^ν. But such a case has already been excluded (in Game $\mathbf{G}_2$). A similar remark holds on the session key:

$$\Pr[\mathsf{A}_3] \le \frac{q_s}{2^{l_1}} \qquad \Pr[\mathsf{SwA}_3] = \frac{1}{2} \ . \tag{5}$$

When collisions of the partial transcripts have been excluded, the event $\mathsf{AskH1}$ can be split in three disjoint sub-cases:

- $\mathsf{AskH1\text{-}Passive}_3$: 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;
- $\mathsf{AskH1\text{-}WithSG}_3$: 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;
- $\mathsf{AskH1\text{-}WithS}_3$: 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 $\mathbf{G}_4$ (Introduction of Diffie-Hellman instance): In order to evaluate the above events, we introduce a random Diffie-Hellman instance (P, Q) (where both P and Q are generators of $\mathbb{G}_p$. Otherwise, the Diffie-Hellman problem is easy.) We first modify the simulation of the party SG for the element Q.

▶ **Rule SG1$^{(4)}$**

Set $h \leftarrow Q$ and compute $W_i \equiv Q^{\omega_i}$, for i $(1 \le i \le n)$, where $\omega_i \overset{R}{\leftarrow} \mathbb{Z}_p^\star$.

By the isomorphic property of $\mathbb{G}_p$, the new W_i is perfectly indistinguishable from before since there exists a unique discrete logarithm for each W_i. We also introduce the other part P of the Diffie-Hellman instance in the simulation of the party S.

▶ **Rule S1**$^{(4)}$

Choose random elements $y \overset{R}{\leftarrow} \mathbb{Z}_p^\star$ and $\mathsf{T} \overset{R}{\leftarrow} \mathbb{G}_p$, and compute $Y \equiv P^y$.

It would let the probabilities unchanged, but note that we excluded the cases $W_i \equiv 1$ and $Y \equiv 1$.

Game $\mathbf{G}_5$ (Probability of AskH): It is now possible to evaluate the probability of the event AskH (or more precisely, the sub-cases). Indeed, one can see that the password is never used during the simulation. It doest not need to be chosen in advance, but at the very end only. Then, an information-theoretic analysis can be done which simply uses cardinalities of some sets.

To this aim, we first cancel a few more games, involved in a communication between an instance S^ν and either the adversary or an instance SG^μ. That is, for some pairs $(t, \{X_i^*\}, Y, \{Z_j\})$ there are two events (which are denoted GuessS_5 and CollW_5) to be explained below.

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

The event GuessS_5 is to guess S and it is clearly bounded by:

$$\Pr[\mathsf{GuessS}_5] \leq \frac{q_s + q_e}{|\mathbb{G}_p|} \ . \tag{6}$$

The CollW_5 is an event that there are two distinct elements S where the tuple $(t, \{X_i^*\}, Y, \{Z_j\}, \mathsf{S})$ is in $\Lambda_\mathcal{H}$ and S is the secret recovered from t shares $\mathsf{S}_j = Z_j \bigoplus \mathcal{G}(j, K_j)$. Here we claim the following:
Claim. For any pair $(t, \{X_i^*\}, Y, \{Z_j\})$ involved in a communication with an instance S^ν, there are two distinct elements W_j, such that $(j, \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_j, Y))$ is in $\Lambda_\mathcal{G}$, granted that two distinct elements S exist where the tuple $(t, \{X_i^*\}, Y, \{Z_j\}, \mathsf{S})$ is in $\Lambda_\mathcal{H}$ and S is the secret recovered from t shares $\mathsf{S}_j = Z_j \bigoplus \mathcal{G}(j, K_j)$. *Proof.* Let $U_j = \mathcal{G}(j, K_j)$. Note that U_j is effectively a one-time pad for Z_j and $\{Z_j\}$ is controlled by the simulator. That means, if there are two distinct elements S, there are also two distinct elements W_j for at least one j, such that the tuple $(j, \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_j, Y))$ is in $\Lambda_\mathcal{G}$, since we have already excluded the random guess on S above and the collision on the output of $\mathcal{G}$ at Game $\mathbf{G}_2$. □

Now the event CollW_5 can be upper-bounded by the following lemma:

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

$$\Pr[\mathsf{CollW}_5] \leq \frac{nq_g^2}{2} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + \tau_e) \ . \tag{7}$$

Proof. We assume that there exist $(\{X_j^*\}, Y \equiv P^y) \in \mathbb{G}_p^{n+1}$ involved in a communication with an instance S^ν, and two elements $W_{j0} \equiv Q^{\omega_{j0}}$ and $W_{j1} \equiv Q^{\omega_{j1}}$, for each j, such that the tuple $(j, K_{jm} \overset{\mathsf{def}}{=} \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_{jm}, Y))$ is in $\Lambda_\mathcal{G}$, for $m = 0, 1$. Then,

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

As a consequence,

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

and thus $\mathsf{CDH}_{g,\mathbb{G}_p}(P, Q) = (K_{j1}/K_{j0})^\psi$ where ψ is the inverse of $y(\omega_{j0} - \omega_{j1})$ 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 $\mathcal{G}$ for any j, one can get the above result (the upper-bound is obtained when $t = 2$). □

In order to conclude the proof, let us study separately the three sub-cases of $\mathsf{AskH1}$ and then $\mathsf{AskH2w1}$ (keeping in mind the absence of several kinds of collisions: for partial transcripts, for outputs of $\mathcal{F}$ and $\mathcal{G}$, and for W_j in $\mathcal{G}$-queries):

- $\mathsf{AskH1\text{-}Passive}$: 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* $(\{X_j^*\}, Y) \in \mathbb{G}_p^{n+1}$*, involved in a passive transcript, there is an element* W_j *such that* $(j, K_j = \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_j, Y))$ *is in* $\Lambda_{\mathcal{G}}$*, one can solve the computational Diffie-Hellman problem:*

  $$\Pr[\mathsf{AskH1\text{-}Passive}_5] \leq \frac{nq_g}{2} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + 2\tau_e) \ . \tag{8}$$

  Proof. We assume that there exist $(\{X_j^* \equiv g^{x_j}\}, Y \equiv P^y) \in \mathbb{G}_p^{n+1}$ involved in a passive transcript and $W_j \equiv Q^{\omega_j}$, for each j, such that the tuple $(j, K_j = \mathsf{CDH}_{g,\mathbb{G}_p}(X_j^*/W_j, Y))$ is in $\Lambda_{\mathcal{G}}$. As above,

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

  As a consequence, $\mathsf{CDH}_{g,\mathbb{G}_p}(P, Q) = (K_j/P^{x_j y})^{\psi}$ where ψ is the inverse of $-y\omega_j$ in $\mathbb{Z}_p^{\star}$. The latter exists since we have excluded the cases where $y = 0$ and $w_j = 0$. By guessing the query asked to the $\mathcal{G}$ for any j, one can get the above result (the upper-bound is also obtained when $t = 2$). □

- $\mathsf{AskH1\text{-}WithSG}$: This corresponds to an attack where the adversary tries to impersonate S to SG (break unilateral authentication). But each authenticator sent by the adversary has been computed with at least two $\omega_j = \mathcal{F}(j, pw_{C_j})$ since $t > 1$ and SG can check the degree of $f(x)$. The maximal probability of the adversary can be obtained when $t = 2$ (see Section 6 for more details):

  $$\Pr[\mathsf{AskH1\text{-}WithSG}_5] \leq \left\lceil \frac{n}{2} \right\rceil^2 \frac{q_s}{N(N-1)} \ . \tag{9}$$

- $\mathsf{AskH1\text{-}WithS}$: 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 $W_i \equiv h^{\mathcal{F}(i, pw_{C_i})}$ the corresponding tuple is in $\Lambda_{\mathcal{G}}$: the probability for the adversary over a random password is as above:

  $$\Pr[\mathsf{AskH1\text{-}WithS}_5] \leq \left\lceil \frac{n}{2} \right\rceil^2 \frac{q_s}{N(N-1)} \ . \tag{10}$$

About $\mathsf{AskH2w1}$ (when the above three events did not happen), it means that only executions with an instance of S (and either SG or the adversary) may lead to acceptance. Exactly the same analysis as for $\mathsf{AskH1\text{-}Passive}$ and $\mathsf{AskH1\text{-}WithS}$ leads to

$$\Pr[\mathsf{AskH2w1}_5] \leq \frac{nq_g}{2} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + 2\tau_e) + \left\lceil \frac{n}{2} \right\rceil^2 \frac{q_s}{N(N-1)} \ . \tag{11}$$

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

$$\Pr[\mathsf{AskH}_5] \leq \left\lceil \frac{n}{2} \right\rceil^2 \frac{3q_s}{N(N-1)} + nq_g \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + 2\tau_e) \ . \tag{12}$$

Combining equation (4), (5), (6), (7) and (12), one gets either

$$\Pr[\mathsf{A}_0] \leq \frac{q_s}{2^{l_1}} + \Delta \qquad \Pr[\mathsf{SwA}_0] = \frac{1}{2} + \Delta \ , \tag{13}$$

where

$$\begin{aligned} \Delta &\leq \left\lceil \frac{n}{2} \right\rceil^2 \frac{3q_s}{N(N-1)} + nq_g \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}}(t_1 + 2\tau_e) + \frac{nq_g^2}{2} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + \tau_e) \\ &\quad + \frac{(q_e + q_s)^2}{2|\mathbb{G}_p|^2} \times \left(1 + \frac{1}{|\mathbb{G}_p|}\right) + \frac{q_f^2}{2p} + \frac{q_g^2 + 2(q_s + q_e)}{2|\mathbb{G}_p|} \\ &\leq \frac{3(n+1)^2 q_s}{4N(N-1)} + \frac{3nq_g^2}{2} \times \mathsf{Succ}^{\mathsf{cdh}}_{g,\mathbb{G}_p}(t_1 + 3\tau_e) \\ &\quad + \frac{(q_e + q_s)^2}{|\mathbb{G}_p|^2} + \frac{q_f^2}{2p} + \frac{q_g^2 + 2(q_s + q_e)}{2|\mathbb{G}_p|} \ . \end{aligned} \tag{14}$$

One can get the result as desired by noting that $\Pr[\mathsf{S}_0] \le \Pr[\mathsf{SwA}_0] + \Pr[\mathsf{A}_0]$. $\square$

B A Countermeasure to Two Attacks of [25]

In this section, we give a countermeasure of the TAP ($t \ge 2$) protocol against both the impersonation and off-line dictionary attacks, shown in [25]. The idea is simple in that we just add a tag for each client to the second message from the server to the subgroup (see below).

Step 2'

- **2.1'** same as **Step 2.1**
- **2.2'** same as **Step 2.2** except the following addition (shown in bold): server S generates a tag $\mathbf{T_j} \leftarrow \mathcal{H}_3(C_j||j||S||t||\{X_i^*\}_{1\le i\le n}||Y||\{Z_j\}_{1\le j\le n}||\mathsf{S}_j||K_j)$, for j ($1 \le j \le n$), where $\mathcal{H}_3$ is a secure one-way hash function.
- **2.3'** same as **Step 2.3** except the following changes (shown in bold): Also, server S generates an authenticator $V_S \leftarrow \mathcal{H}_1(C||S||t||\{X_i^*\}_{1\le i\le n}||Y||\{(Z_j, \mathbf{T_j})\}_{1\le j\le n}||\mathsf{S})$ and a session key $SK \leftarrow \mathcal{H}_2(C||S||t||\{X_i^*\}_{1\le i\le n}||Y||\{(Z_j, \mathbf{T_j})\}_{1\le j\le n}||\mathsf{S})$. Then, the server sends its identity S, the Diffie-Hellman public value Y, $\{(Z_j, \mathbf{T_j})\}_{1\le j\le n}$ and the authenticator V_S to subgroup SG.

Step 3'

- **3.1'** same as **Step 3.1** except the following addition (shown in bold): if $\mathbf{T_i} \ne \mathcal{H}_3(C_i||i||S||t||\{X_i^*\}_{1\le i\le n}||Y||\{Z_j\}_{1\le j\le n}||\mathsf{S}_i||K_i)$, each client C_i in the subgroup SG terminates the protocol. Otherwise, proceed to **Step 3.2'**.
- **3.2'** same as **Step 3.2** with the obvious changes in verifying V_S and generating SK. Note that subgroup SG should reconstruct and share S' *securely*[6] from the t shares S_i.

C An Attack on the D-NAPAKE Protocol of [25]

In this section, we show an attack on the D-NAPAKE protocol of [25] where only one legitimate client can impersonate any subgroup of clients to the server. In other words, the D-NAPAKE protocol is **NOT** a threshold anonymous PAKE protocol.

C.1 The D-NAPAKE Protocol

Here, we describe the D-NAPAKE protocol of [25] (see Fig. 4). The main idea of the D-NAPAKE protocol is that, after sharing a secret based on SPEKE [13], the subgroup and the server run a masked sequential Diffie-Hellman protocol in a threshold secret sharing manner.

Let $\mathbb{G} = \langle g \rangle$ be a finite, cyclic group of prime order q. Let $\mathcal{G} : \{0,1\}^* \rightarrow \mathbb{G}$ be a full-domain hash function, and $\mathcal{H}_0, \mathcal{H}_1 : \{0,1\}^* \rightarrow \{0,1\}^l$ be two random hash functions where l is the security parameter. Let pw_i be a password shared between the client C_i 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 $r_S \overset{R}{\leftarrow} \mathbb{Z}_q^\star$, 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 subgroup SG.
2. The subgroup SG ($\subset \Gamma$) checks all the values in $\{A_j\}_{1\le j\le n}$ are different one another. If not, subgroup SG aborts the protocol. Otherwise, each client $C_i \in SG$ picks A_i from $\{A_j\}_{1\le j\le n}$, and chooses two random numbers $(r_i, x_i) \overset{R}{\leftarrow} \mathbb{Z}_q^\star$. 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 $y \overset{R}{\leftarrow} \mathbb{Z}_q^\star$ 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 [16] 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 $Trans = \Gamma||S||\{A_j\}_{1\le j\le n}||t||\{B_{i1}, B_{i2}\}_{1\le i\le n}||\{K_j\}_{1\le j\le n}$. Finally, server S sends $(\{K_j\}_{1\le j\le t}, Auth_S)$ to subgroup SG.

---

[6] With the use of secure channels

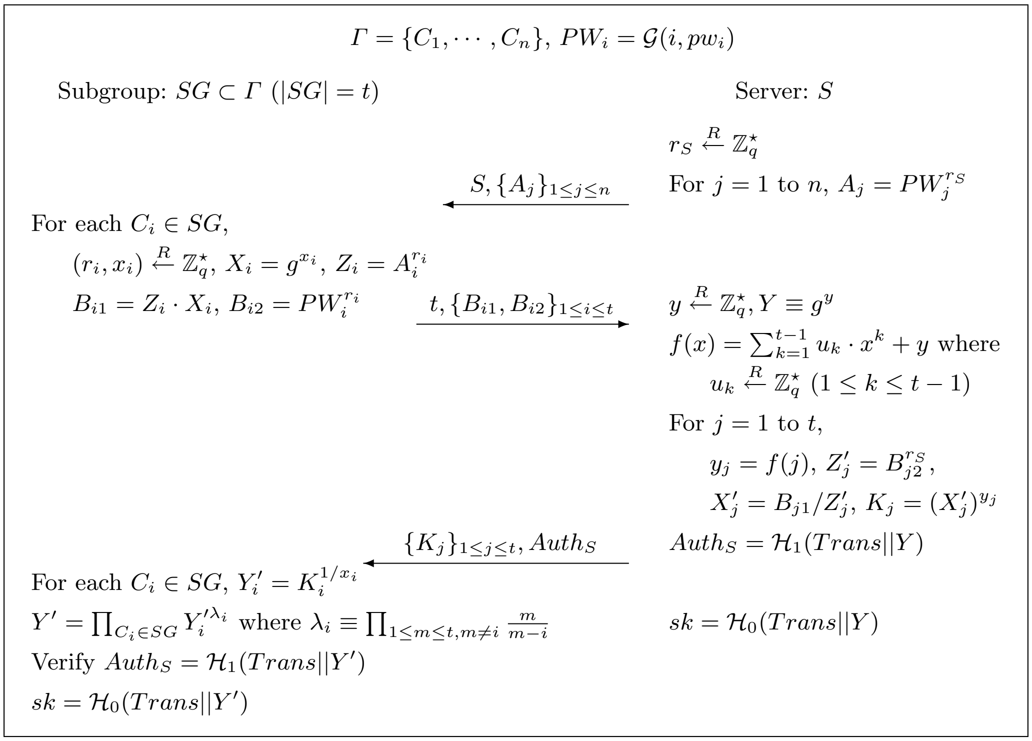

Fig. 4. The D-NAPAKE protocol [25] where the threshold $t \geq 2$ and $Trans = \Gamma||S||\{A_j\}_{1\leq j\leq n}||t||\{B_{i1}, B_{i2}\}_{1\leq i\leq n}||\{K_j\}_{1\leq j\leq n}$

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 abots the protocol. Otherwise, subgroup SG computes a session key $sk = \mathcal{H}_0(Trans||Y')$ and accepts it.

C.2 The Attack

Now, we are ready to show an attack on the D-NAPAKE protocol of [25]. W.l.o.g., client $C_l \in \Gamma$, who is sharing pw_l with server S, is trying to impersonate any subgroup SG ($t \geq 2$) of clients to server S. Note that Yang and Zhang [25] proposed the D-NAPAKE protocol as a threshold version of NAPAKE protocol so that server S should authenticate any subgroup SG ($t \geq 2$) of clients anonymously.

1. same as 1. of Section C.1
2. The client $C_l \in \Gamma$ picks A_l from $\{A_j\}_{1\leq j\leq n}$, and chooses $2t$ random numbers $\{(r_i, x_i)\}_{1\leq i\leq t} \xleftarrow{R} \mathbb{Z}_q^\star$. For i $(1 \leq i \leq t)$, the 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 $(t, \{B_{i1}, B_{i2}\}_{1\leq i\leq t})$ to server S where the threshold t $(t \geq 2)$.
3. same as 3. of Section C.1
4. The client $C_l \in \Gamma$ computes $Y'_i = K_i^{1/x_i}$ for i $(1 \leq i \leq t)$. Then, the client C_l recovers Y' from t Y'_i values by Lagrange interpolation. Finally, the client C_l shares a session key $sk = \mathcal{H}_0(Trans||Y')$ with the server S because $Y' = Y$.

CORRECTNESS OF THE ATTACK. It is enough to show $K_j = (g^{x_i})^{y_j}$ for $i = j$:

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