Security Proof of AugPAKE

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

Abstract. In this paper, we show that the AugPAKE protocol [16] provides the semantic security of session keys under the strong Diffie-Hellman (SDH) assumption in the random oracle model.

Key words: PAKE, on-line/off-line dictionary attacks, provable security, SDH, random oracle

1 Introduction

In 1976, Diffie and Hellman [6] published their seminal paper that introduced how to share a secret over public networks (so-called, the Diffie-Hellman key exchange). The idea of their protocol became foundation of the modern public key cryptography. Since then, many researchers have tried to design secure cryptographic algorithms/protocols for realizing secure channels. These algorithms/protocols are necessary because application-oriented protocols (*e.g.*, web mail, Internet banking/shopping, ftp) are frequently developed assuming the existence of such secure channels. In the 2-party setting (*i.e.*, client and 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). An AKE protocol is usually designed to be secure against active attacks by adding authentication in any form to the underlying key exchange protocol. Note that the Diffie-Hellman protocol [6] is only secure against an adversary who can eavesdrop communications between the parties.

For authentication, human-memorable passwords (*e.g.*, 4-digit pin-code) are commonly used rather than high-entropy keys because of their convenience in use and wide deployment in practice. Many password-based AKE protocols have been extensively investigated in the literature 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 major attacks on passwords: on-line and off-line dictionary attacks. Let us take for example a simple challenge-response password authentication protocol where a client and a server share a password pw. In the protocol, the server first 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 one-way hash function. After receiving r, the server authenticates the client if $r = \mathcal{H}(c, pw)$. 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, after eavesdropping the pair (c, r) an adversary can find out the correct password pw with off-line dictionary attacks by trying all password candidates pw' until it satisfies $r = \mathcal{H}(c, pw')$. This attack is possible since passwords are chosen from a relatively-small dictionary that allows the exhaustive searches. 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.

Due to the existence of off-line dictionary attacks on passwords, it is not trivial at all to design a secure key exchange protocol between the parties, who share a low-entropy password only. In [2], Bellovin

and Merritt brought forth this problem and proposed a suit of solutions, called Encrypted Key Exchange (EKE). Though some attacks (*e.g.*, [13]) are found, these EKE protocols are good examples 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). Since then, a number of password-only AKE protocols (see [8, 11]) have been proposed in the literature with the name of Password-Authenticated Key Exchange (PAKE). Among them, some PAKE protocols have been standardized in IEEE P1363.2 [8], ISO/IEC [9], IETF [17] and ITU-T [10].

In this paper, we show that the AugPAKE protocol [16] provides the semantic security of session keys under the strong Diffie-Hellman (SDH) assumption in the random oracle model.

2 Preliminary

2.1 Notation

Here, we explain some notation to be used throughout this paper. Let $\mathbb{G}$ be a finite, cyclic group of prime order q (*i.e.*, residues modulo p where $p = aq + 1$ is a prime and a is an integer) and g be a generator of $\mathbb{G}$ where the group operation is denoted multiplicatively. For computational efficiency, we recommend to use a secure prime p such that $p = 2qp_1p_2 \cdots p_o + 1$ where $o \geq 1$ and, for i $(1 \leq i \leq o)$, each factor p_i is also a prime comparable to q in size.[3] Let l be the security parameter for $\mathbb{G}$ so that $l = \lceil \log q \rceil$. The (p, q, g) are given as public parameters. In the aftermath, all the subsequent arithmetic operations are performed in modulo p unless otherwise stated.

Let k be the security parameter for hash functions. Let N be a dictionary size of passwords. Let $\{0,1\}^\star$ denote the set of finite binary strings and $\{0,1\}^k$ the set of binary strings of length k. Let $A \| B$ be the concatenation of bit strings of A and B in $\{0,1\}^\star$. If D is a set, then $d \stackrel{R}{\leftarrow} D$ indicates the process of selecting d at random and uniformly over D. We use two different hash functions $(\mathcal{H}, \widetilde{\mathcal{H}})$ and $\mathcal{H}_j$, for $j = 1, 2, 3$, where $(\mathcal{H}, \widetilde{\mathcal{H}}) : \{0,1\}^\star \rightarrow \mathbb{Z}_q^\star$ and $\mathcal{H}_j : \{0,1\}^\star \rightarrow \{0,1\}^k$. The $(\mathcal{H}, \widetilde{\mathcal{H}})$ and $\mathcal{H}_j$ are implemented with secure one-way hash functions (*e.g.*, SHA-2 family). Let C and S be the identities of client and server, respectively, with each identity $\mathrm{ID} \in \{0,1\}^\star$.

Let $\mathbb{N}$ (resp., $\mathbb{R}^+$) be the set of natural (resp., positive real) numbers. We say that a function $\varepsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ is *negligible* if and only if for every polynomial $f(n)$ there exists an $n_0 \in \mathbb{N}$ such that for all $n > n_0$, $\varepsilon(n) \leq 1/f(n)$.

2.2 The Formal Model for PAKE

Security Model [3] In an (augmented) PAKE protocol P, there are a fixed number of parties who can be either a client or a server. For simplicity, we only consider two different parties C and S who share a low-entropy secret, chosen uniformly[4] from a small dictionary $\mathbb{D}_{\mathsf{password}}$.[5] We fix the cardinality of $\mathbb{D}_{\mathsf{password}}$ to N. Each of C and S may have several instances, called oracles involved in distinct, possibly concurrent, executions of P. We denote C (resp., S) instances by C^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 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:

- $\mathsf{Execute}(C^\mu, S^\nu)$: This query models passive attacks, where the adversary gets access to honest executions of P between the instances C^μ and S^ν by eavesdropping. Note that the Execute-query is important in order to evaluate on-line/off-line dictionary attacks in P.
- $\mathsf{Send}(I, m)$: 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 $\mathsf{Send}(C^\mu, \mathtt{Start})$ initializes the protocol, and then the adversary receives the first message.

[3] Alternatively, one may use a safe prime p such that $p = 2q + 1$ with sacrificing computational efficiency.
[4] It can be easily extended to the case of non-uniform password distributions (see [12]).
[5] In an augmented PAKE protocol, client C remembers his/her password pw and server S holds its verification data (not pw itself) derived from the password.

- $\mathsf{Reveal}(I)$: This query handles misuse of the session key [7] 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 $\mathcal{A}$.
- $\mathsf{Test}(I)$: This query is used to measure how much the adversary can obtain information about the session key. The Test-query can be asked at most once by the adversary $\mathcal{A}$ and is only available to $\mathcal{A}$ if the instance 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 $b = 0$.

We consider an instance I in the above Test-query. Let I' be a partnered instance of I.[6] The instance I is *fresh* if the following conditions are satisfied: (1) I has accepted (*i.e.*, holding a session key SK) and therefore it has an SID; and (2) neither $\mathsf{Reveal}(I)$ nor $\mathsf{Reveal}(I')$ has been asked by $\mathcal{A}$.

Security Notion 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 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. 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.

2.3 Computational Assumption

Here, we define the strong Diffie-Hellman (SDH) assumption [1].

Definition 1. (v-SDH Problem) *Let $\mathbb{G}$ be a finite cyclic group of prime order q with g as a generator, described above. A v-$\mathsf{SDH}_{g,\mathbb{G}}$ attacker is a probabilistic polynomial time (PPT) algorithm $\mathcal{B}$ running in time t such that its success probability $\mathsf{Succ}^{\mathsf{vsdh}}_{g,\mathbb{G}}$, given as input $(v+1)$-tuple of elements $\left(g, g^u, g^{(u^2)}, \cdots, g^{(u^v)}\right)$ to output a pair $\left(c, g^{1/(u+c)}\right)$ for a freely chosen value $c \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$, is greater than ε. We denote by $\mathsf{Succ}^{\mathsf{vsdh}}_{g,\mathbb{G}}(t)$ the maximal success probability over every adversaries running within time t. The v-SDH assumption states that $\mathsf{Succ}^{\mathsf{vsdh}}_{g,\mathbb{G}}(t) \leq \varepsilon$ for any t/ε not too large.*

For $v = 1$, we have the following definition:

Definition 2. (1-SDH Problem) *Let $\mathbb{G}$ be a finite cyclic group of prime order q with g as a generator, described above. A 1-$\mathsf{SDH}_{g,\mathbb{G}}$ attacker is a probabilistic polynomial time (PPT) algorithm $\mathcal{B}$ running in time t such that its success probability $\mathsf{Succ}^{\mathsf{1sdh}}_{g,\mathbb{G}}$, given as input 2-tuple of elements (g, g^u) to output a pair $\left(c, g^{1/(u+c)}\right)$ for a freely chosen value $c \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$, is greater than ε. We denote by $\mathsf{Succ}^{\mathsf{1sdh}}_{g,\mathbb{G}}(t)$ the maximal success probability over every adversaries running within time t. The 1-SDH assumption states that $\mathsf{Succ}^{\mathsf{1sdh}}_{g,\mathbb{G}}(t) \leq \varepsilon$ for any t/ε not too large.*

3 The AugPAKE Protocol

In this section, we describe the AugPAKE protocol [16] with the extremely-low computational costs. In fact, the computational efficiency of the AugPAKE protocol is almost same as the plain Diffie-Hellman protocol [6] which does not provide authentication at all.

In the initialization phase of the AugPAKE protocol, client C registers his/her password verification data $W \equiv g^{pw}$ securely to server S where pw is the client's password. In other words, client C remembers the password pw only and server S holds the W.

In the actual execution of the AugPAKE protocol, a pair of client and server can share an authenticated session key over insecure networks (see Fig. 1). Of course, X and Y should not be $0, \pm 1 \bmod p$ in order to avoid trivial attacks.

[6] Formally, the instances I and I' are partnered if they have the same session id (SID) which is defined as the concatenation of all the exchanged messages sent and received by an instance. Refer to Section 3 of [3] for more details.

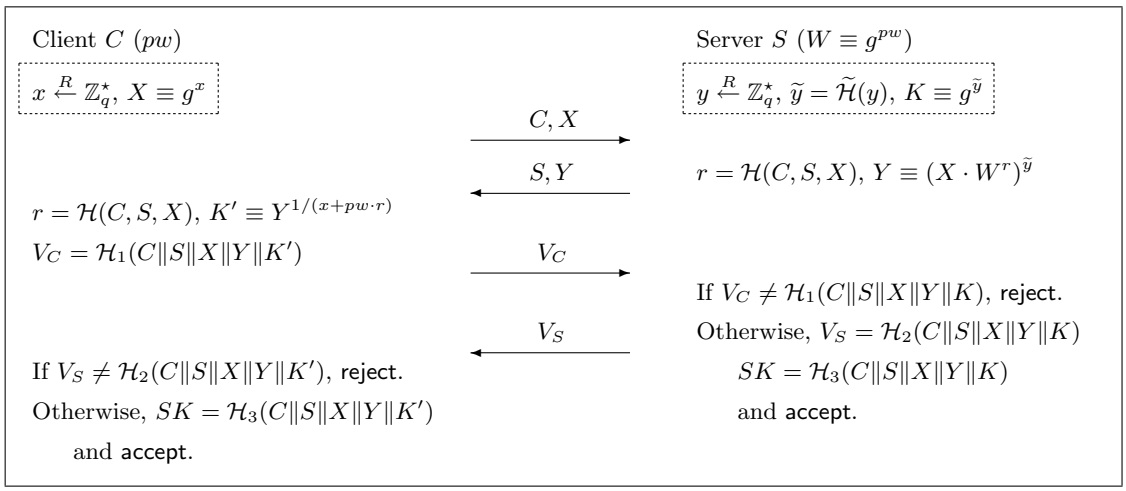

Fig. 1. The AugPAKE protocol [16] where the dashed boxes represent pre-computable

Step 0 (Pre-computation): At first, client C chooses a random element x from $\mathbb{Z}_q^\star$ and computes its Diffie-Hellman public value $X \equiv g^x$. Similarly, server S chooses a random element y from $\mathbb{Z}_q^\star$, and computes $\widetilde{y} = \widetilde{\mathcal{H}}(y)$ and $K \equiv g^{\widetilde{y}}$. These computations can be done off-line.

Step 1: The client C sends the first message (C, X) to server S.
Step 2: After receiving (C, X), server S computes $Y \equiv (X \cdot W^r)^{\widetilde{y}}$ where $r = \mathcal{H}(C, S, X)$. Then, server S sends the second message (S, Y) to client C.
Step 3: After receiving (S, Y), client C computes $K' \equiv Y^{1/(x+pw \cdot r)}$ where $r = \mathcal{H}(C, S, X)$. Also, the client generates an authenticator $V_C = \mathcal{H}_1(C\|S\|X\|Y\|K')$ and then sends the third message V_C to server S.
Step 4: After receiving V_C, server S checks the correctness of V_C. If V_C is not equal to $\mathcal{H}_1(C\|S\|X\|Y\|K)$, server S terminates the protocol with an error. Otherwise, the server generates its authenticator $V_S = \mathcal{H}_2(C\|S\|X\|Y\|K)$ and a session key $SK = \mathcal{H}_3(C\|S\|X\|Y\|K)$. Then, server S sends the fourth message V_S to client C.
Step 5: After receiving V_S, client C checks the correctness of V_S. If V_S is not equal to $\mathcal{H}_2(C\|S\|X\|Y\|K')$, client C terminates the protocol with an error. Otherwise, the client generates a session key $SK = \mathcal{H}_3(C\|S\|X\|Y\|K')$.

4 Security Proof

In this section, we show that the AugPAKE protocol [16] provides the AKE security under the strong Diffie-Hellman (SDH) assumption in the random oracle model.

Theorem 1. (AKE Security) *Let P be the* AugPAKE *protocol of Fig. 1 where passwords are independently chosen from a dictionary of size N. 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 (*Execute*-queries) and asking q_{hashH}, $q_{\mathsf{hash}\widetilde{\mathsf{H}}}$ and q_{hashH_j} hash queries to $\mathcal{H}$, $\widetilde{\mathcal{H}}$ and any $\mathcal{H}_j$, for $j = 1, 2, 3$, respectively,*

$$\mathsf{Adv}_P^{\mathsf{ake}}(\mathcal{A}) \leq \frac{6(q_{\mathsf{sendC}} + q_{\mathsf{sendS}})}{N} + \frac{(q_{\mathsf{execute}} + q_{\mathsf{send}})^2}{|\mathbb{G}|} + \frac{q_{\mathsf{sendC}} + q_{\mathsf{sendS}}}{2^{k-1}} + \frac{q'^2_{\mathsf{hashH}} + q^2_{\mathsf{hash}\widetilde{\mathsf{H}}}}{q} + \left(2N^2 \cdot q_{\mathsf{hash}\widetilde{\mathsf{H}}} + 6N\right) \times \mathsf{Succ}^{\mathsf{1sdh}}_{g,\mathbb{G}}(t + \tau_e) \quad (1)$$

where (1) q_{sendC} (resp., q_{sendS}) is the number of Send*-queries to C (resp., S) instance, (2) $q_{\mathsf{send}} \leq q_{\mathsf{sendC}} + q_{\mathsf{sendS}}$ and $q'_{\mathsf{hashH}} \leq q_{\mathsf{hashH}} + q_{\mathsf{hashH}_j}$, (3) k is the output size of hash functions $\mathcal{H}_j$, and (4) τ_e denotes the computational time for a modular exponentiation in $\mathbb{G}$.*

This theorem shows that the AugPAKE protocol is AKE-secure against off-line dictionary attacks since the advantage of the adversary essentially grows with the ration of interactions (number of Send-queries) to the number of passwords.

In the proof, we incrementally define a sequence of games, starting at the real game $\mathbf{G}_0$ and ending up at $\mathbf{G}_5$. We use Shoup's difference lemma [14, 15] 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** $\mathbf{G}_i$. Since the proof is in the random oracle model [4], the public parameters (*e.g.*, (p, q, g)) do not include the descriptions of the hash functions $(\mathcal{H}, \widetilde{\mathcal{H}})$ and $\mathcal{H}_j$, for $j = 1, 2, 3$.

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

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

Game $\mathbf{G}_1$: In this game, we simulate the hash oracles ($(\mathcal{H}, \widetilde{\mathcal{H}})$ and $\mathcal{H}_j$, for $j = 1, 2, 3$) as usual by maintaining these hash lists $List_{\mathcal{H}}$, $List_{\widetilde{\mathcal{H}}}$ and $List_{\mathcal{H}_j}$. Additionally, we simulate another hash functions $\mathcal{H}'_j : \{0,1\}^\star \rightarrow \{0,1\}^k$, for $j = 1, 2, 3$, with hash lists $List_{\mathcal{H}'_j}$ which are private oracles to the simulator.

- For a hash-query $\mathcal{H}(Q)$ (resp., $\widetilde{\mathcal{H}}(Q)$), such that a record (Q, r) appears in $List_{\mathcal{H}}$ (resp., $List_{\widetilde{\mathcal{H}}}$), the answer is r. Otherwise, one chooses a random element $r \xleftarrow{R} \mathbb{Z}_q^\star$, answers with it, and adds the record (Q, r) to $List_{\mathcal{H}}$ (resp., $List_{\widetilde{\mathcal{H}}}$).
- For a hash-query $\mathcal{H}_j(Q)$ (resp., $\mathcal{H}'_j(Q)$), such that a record (j, Q, r) appears in $List_{\mathcal{H}_j}$ (resp., $List_{\mathcal{H}'_j}$), the answer is r. Otherwise, one chooses a random element $r \xleftarrow{R} \{0,1\}^k$, answers with it, and adds the record (j, Q, r) to $List_{\mathcal{H}_j}$ (resp., $List_{\mathcal{H}'_j}$).

We also simulate all the instances for all queries (*i.e.*, $\mathsf{Execute}(C^\mu, S^\nu)$, $\mathsf{Send}(C^\mu, \mathtt{Start})$, $\mathsf{Send}(C^\mu, (S, Y))$, $\mathsf{Send}(C^\mu, V_S)$, $\mathsf{Send}(S^\nu, (C, X))$, $\mathsf{Send}(S^\nu, V_C)$, $\mathsf{Reveal}(I)$ and $\mathsf{Test}(I)$-queries) asked by an adversary. This is done exactly as the real instances would do. From the above simulation, it is clear that

$$\Pr[\mathsf{S}_1] \approx \Pr[\mathsf{S}_0] \ . \tag{3}$$

Game $\mathbf{G}_2$: In this game, we first forward any hash-query $\mathcal{H}_j(Q)$ to $\mathcal{H}_j$ oracle to $\mathcal{H}$ oracle as follows: The query Q is parsed as $Q = C\|S\|X\|Y\|K$, and then we forward a hash-query $\mathcal{H}(C, S, X)$ to $\mathcal{H}$ oracle. So, the number of $\mathcal{H}$-queries becomes $q'_{\mathsf{hashH}} \leq q_{\mathsf{hashH}} + q_{\mathsf{hashH}_j}$. Let Coll_2 be an event where some collisions are going to happen.

- Collisions on the partial transcript $((C, X), (S, Y))$: an adversary tries to find out a pair (X, Y), coinciding with the transcript of the test instance (asked in the Test-query), and then obtain the corresponding session key using the Reveal-query. However, at least one party involves with the transcript, and thus one of X and Y is uniformly distributed.
- Collisions on the output of $\mathcal{H}$ and $\widetilde{\mathcal{H}}$

These probabilities are upper-bounded by the birthday paradox:

$$\Pr[\mathsf{Coll}_2] \leq \frac{(q_{\mathsf{execute}} + q_{\mathsf{send}})^2}{2|\mathbb{G}|} + \frac{q'^2_{\mathsf{hashH}} + q^2_{\mathsf{hash}\widetilde{\mathsf{H}}}}{2q} \ . \tag{4}$$

Game $\mathbf{G}_3$: In this game, we make the authenticators and the session key unpredictable to an adversary by using the private oracles $\mathcal{H}'_j$ instead of $\mathcal{H}_j$, for $j = 1, 2, 3$. Specifically, we compute $V_C = \mathcal{H}'_1(C\|S\|X\|Y)$, $V_S = \mathcal{H}'_2(C\|S\|X\|Y)$ and $SK_{C/S} = \mathcal{H}'_3(C\|S\|X\|Y)$. Note that we do not need to use K in the computation of $\mathcal{H}'_j$ because the resultant values are completely independent from $\mathcal{H}_j$. Subsequently, we change the simulation of Send-oracle as follows: For a Send-query $\mathsf{Send}(C^\mu, (S, Y))$, we do not compute anything and just respond with V_C. We can also simplify the generation of Y using the homomorphic property of $\mathbb{G}$: For a Send-query $\mathsf{Send}(S^\nu, (C, X))$, we choose a random element $y \xleftarrow{R} \mathbb{Z}_q^\star$, compute $Y \equiv g^y$, and respond with (S, Y).

The games $\mathbf{G}_3$ and $\mathbf{G}_2$ are indistinguishable unless some specific hash queries are asked, denoted by event $\mathsf{AskH}_3 = \mathsf{AskH1}_3 \vee \mathsf{AskH2w1}_3 \vee \mathsf{AskH3w12}_3$:

- $\mathsf{AskH1}_3$: $\mathcal{H}_1(C\|S\|X\|Y\|K)$ has been queried by $\mathcal{A}$ to $\mathcal{H}_1$ for some execution transcripts $((C, X), (S, Y))$;

- $\mathsf{AskH2w1}_3$: $\mathcal{H}_2(C\|S\|X\|Y\|K)$ has been queried by $\mathcal{A}$ to $\mathcal{H}_2$ for some execution transcripts $((C,X),(S,Y))$ but event $\mathsf{AskH1}_3$ did not happen;
- $\mathsf{AskH3w12}_3$: $\mathcal{H}_3(C\|S\|X\|Y\|K)$ has been queried by $\mathcal{A}$ to $\mathcal{H}_3$ for some execution transcripts $((C,X),(S,Y))$, where one of the parties has accepted, but events $\mathsf{AskH1}_3$ and $\mathsf{AskH2w1}_3$ did not happen;

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

$$\Pr[\mathsf{AskH}_3] \leq \Pr[\mathsf{AskH1}_3] + \Pr[\mathsf{AskH2w1}_3] + \Pr[\mathsf{AskH3w12}_3] \ .$$

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 $((C,X),(S,Y))$ appeared in another session with the real instances C^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 $1/2$:

$$\Pr[\mathsf{S}_3] \leq \frac{q_{\mathsf{sendC}} + q_{\mathsf{sendS}}}{2^k} + \frac{1}{2} \ . \tag{5}$$

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

- $\mathsf{AskH1\text{-}Passive}_3$: the transcript $((C,X),(S,Y))$ comes from an execution between the instances C^μ and S^ν ($\mathsf{Execute}$-queries or forward of Send-queries, relay of part of them). This means that both X and Y have been simulated;
- $\mathsf{AskH1\text{-}WithC}_3$: the execution involved with the instance C^μ, but Y has not been sent by any instance S^ν. This means that X has been simulated, but Y has been produced by the adversary;
- $\mathsf{AskH1\text{-}WithS}_3$: the execution involved with the instance S^ν, but X has not been sent by any instance C^μ. This means that Y has been simulated, but X has been produced by the adversary.

Game $\mathbf{G}_4$: In order to evaluate the above events, we introduce a random SDH instance $U(=g^u)$ into the simulation of the party C: For a Send-query $\mathsf{Send}(C^\mu, \mathtt{Start})$, we choose a random element $r \stackrel{R}{\leftarrow} \mathbb{Z}_q^\star$, compute $X \equiv U^r$, and respond with (C,X). At this moment, we also set r as $r = \mathcal{H}(C,S,X)$.

By the isomorphic property from $\mathbb{G}$ to $\mathbb{G}$, the new X is perfectly indistinguishable from before since there exist a unique discrete logarithm for X. From the above simulation, it is clear that

$$\Pr[\mathsf{AskH}_4] \approx \Pr[\mathsf{AskH}_3] \ . \tag{6}$$

Game $\mathbf{G}_5$: In this game, we bound the probability of event AskH_5 (or, the sub-cases $\mathsf{AskH1}_5$, $\mathsf{AskH2w1}_5$ and $\mathsf{AskH3w12}_5$). In particular, we analyze on-line dictionary attacks by simply using cardinalities of some sets since the password is never used during the simulation. Note that the password verification data W has the same entropy as the password pw (*i.e.*, $W \equiv g^{pw}$).

By using the proof technique [5], we consider an event CollH_5 where, for some pairs $(X,Y) \in \mathbb{G}^2$, involved in a communication between an instance C^μ and either the adversary or an instance S^ν, there are two distinct elements W_i such that the tuple (X,Y,K_i) is in $List_{\mathcal{H}_j}$. This probability is upper-bounded by $\Pr[\mathsf{CollH}_5]$:

$$|\Pr[\mathsf{AskH}_5] - \Pr[\mathsf{AskH}_4]| \leq \Pr[\mathsf{CollH}_5] \ .$$

With the following lemma, the event CollH_5 can be upper-bounded.

Lemma 1. *If for any pair $(X,Y) \in \mathbb{G}^2$, involved in a communication with an instance C^μ, there are two elements W_0 and W_1 such that (X,Y,K_i) is in $List_{\mathcal{H}_j}$, one can solve the strong Diffie-Hellman problem:*

$$\Pr[\mathsf{CollH}_5] \leq \left(N^2 \cdot q_{\mathsf{hash\tilde{H}}}\right) \times \mathsf{Succ}^{\mathsf{1sdh}}_{g,\mathbb{G}}(t + \tau_e) \ . \tag{7}$$

Proof. We prove this lemma by showing the reduction to the strong Diffie-Hellman (SDH) problem when event CollH_5 happens. We assume that there exist $(X \equiv U^r, Y \equiv (X \cdot W_0^r)^{\tilde{y}}) \in \mathbb{G}^2$ involved in a

communication with an instance C^μ, and two elements $W_0 \equiv g^{pw_0}$ and $W_1 \equiv g^{pw_1}$ such that the tuple (X, Y, K_i) is in $List_{\mathcal{H}_j}$, for $i = 0, 1$:

$$K_0 = g^{\widetilde{y}},$$
$$K_1 = Y^{\frac{1}{\log_g(X \cdot W_1^r)}} = Y^{\frac{1}{(u \cdot r + pw_1 \cdot r)}} = g^{\frac{\widetilde{y}(u \cdot r + pw_0 \cdot r)}{(u \cdot r + pw_1 \cdot r)}} = g^{\frac{\widetilde{y}(u + pw_0)}{(u + pw_1)}} .$$

As a consequence,

$$\frac{K_0}{K_1} = \frac{g^{\widetilde{y}}}{g^{\widetilde{y}\left(\frac{u+pw_0}{u+pw_1}\right)}} = g^{\widetilde{y}\left(1 - \frac{u+pw_0}{u+pw_1}\right)} = g^{\widetilde{y}\left(\frac{pw_1 - pw_0}{u+pw_1}\right)} ,$$

and thus the solution to the SDH problem is $\left(pw_1, (K_0/K_1)^\psi\right)$ where ψ is the inverse of $\widetilde{y}(pw_1 - pw_0)$ in $\mathbb{Z}_q^\star$. The latter exists since $W_0 \neq W_1$ and $\widetilde{y} \neq 0$ (*i.e.*, $\widetilde{y} \in \mathbb{Z}_q^\star$). By guessing the query asked to the $\widetilde{\mathcal{H}}$ and two passwords (pw_0, pw_1), we conclude the proof. □

In order to conclude the proof, we separately bound the three sub-cases of $\mathsf{AskH1}_5$, $\mathsf{AskH2w1}_5$ and $\mathsf{AskH3w12}_5$ (keeping in mind the absence of several kinds of collisions).

– $\mathsf{AskH1\text{-}Passive}_5$: About the passive transcripts (in which both X and Y have been simulated), one can state the following lemma:

Lemma 2. *If for any pair $(X, Y) \in \mathbb{G}^2$, involved in a passive transcript, there is an element W such that (X, Y, K) is in $List_{\mathcal{H}_j}$, one can solve the strong Diffie-Hellman problem:*

$$\Pr[\mathsf{AskH1\text{-}Passive}_5] \leq N \times \mathsf{Succ}^{\mathsf{1sdh}}_{g,\mathbb{G}}(t + \tau_e) . \tag{8}$$

Proof. We prove this lemma by showing the reduction to the SDH problem when event $\mathsf{AskH1\text{-}Passive}_5$ happens. We assume that there exist $(X \equiv U^r, Y \equiv g^y) \in \mathbb{G}^2$ involved in a passive transcript and $W \equiv g^{pw}$ such that the tuple (X, Y, K) is in $List_{\mathcal{H}_j}$. As above,

$$K = Y^{\frac{1}{\log_g(X \cdot W^r)}} = g^{\frac{y}{(u \cdot r + pw \cdot r)}} = g^{\frac{y}{r(u+pw)}} .$$

As a consequence, the solution to the SDH problem is $\left(pw, K^\psi\right)$ where ψ is the inverse of y/r in $\mathbb{Z}_q^\star$. The latter exists since $(y, r) \in \left(\mathbb{Z}_q^\star\right)^2$. By guessing the password pw, we conclude the proof. □

– $\mathsf{AskH1\text{-}WithC}_5$: The above Lemma 1 states that for each pair (X, Y) involved in a transcript with an instance C^μ, there is at most one element W such that the corresponding tuple is in $List_{\mathcal{H}_j}$: The probability for the adversary over a random password is obtained by

$$\Pr[\mathsf{AskH1\text{-}WithC}_5] \leq \frac{q_{\mathsf{sendC}}}{N} \tag{9}$$

because the only secret is the password pw.

– $\mathsf{AskH1\text{-}WithS}_5$: This corresponds to an attack where the adversary tries to impersonate C to S. But each authenticator sent by the adversary has been determined from a guessed password. Therefore, the probability for the adversary over a random password is as above:

$$\Pr[\mathsf{AskH1\text{-}WithS}_5] \leq \frac{q_{\mathsf{sendS}}}{N} . \tag{10}$$

About $\mathsf{AskH2w1}_5$ and $\mathsf{AskH3w12}_5$, exactly the same analysis leads to

$$\Pr[\mathsf{AskH2w1}_5] + \Pr[\mathsf{AskH3w12}_5] \leq 2N \times \mathsf{Succ}^{\mathsf{1sdh}}_{g,\mathbb{G}}(t + \tau_e) + \frac{2(q_{\mathsf{sendC}} + q_{\mathsf{sendS}})}{N} . \tag{11}$$

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

$$\Pr[\mathsf{AskH}_5] \leq 3N \times \mathsf{Succ}^{\mathsf{1sdh}}_{g,\mathbb{G}}(t + \tau_e) + \frac{3(q_{\mathsf{sendC}} + q_{\mathsf{sendS}})}{N} . \tag{12}$$

By combining equation (4), (5), (7) and (12), one gets

$$\begin{aligned}\Pr[\mathsf{S}_0] \leq & \frac{\left(q_{\mathsf{execute}} + q_{\mathsf{send}}\right)^2}{2|\mathbb{G}|} + \frac{q'^2_{\mathsf{hashH}} + q^2_{\mathsf{hash\tilde{H}}}}{2q} \\ & + \frac{q_{\mathsf{sendC}} + q_{\mathsf{sendS}}}{2^k} + \frac{1}{2} + \frac{3(q_{\mathsf{sendC}} + q_{\mathsf{sendS}})}{N} \\ & + \left(N^2 \cdot q_{\mathsf{hash\tilde{H}}} + 3N\right) \times \mathsf{Succ}^{\mathsf{1sdh}}_{g,\mathbb{G}}(t + \tau_e) \ . \end{aligned} \tag{13}$$

Finally, the security result as desired is obtained by noting equation (2). □

References

1. D. Boneh and X. Boyen, "Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups", *Journal of Cryptology*, Vol. 21, Issue 2, pp. 149-177, Springer-Verlag, February 2008.
2. 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, IEEE Computer Society, 1992.
3. 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.
4. M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols", In *Proc. of ACM CCS'93*, pp. 62-73, ACM Press, 1993.
5. D. Catalano, D. Pointcheval, and T. Pornin, "Trapdoor Hard-to-Invert Group Isomorphisms and Their Application to Password-based Authentication", *Journal of Cryptology*, Vol. 20, No. 1, pp. 115-149, Springer-Verlag, 2007.
6. W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, Vol. IT-22, No. 6, pp. 644-654, 1976.
7. D. Denning and G. Sacco, "Timestamps in Key Distribution Protocols", *Communications of the ACM*, Vol. 24, pp. 533-536, 1981.
8. IEEE P1363.2, "Draft Standard for Specifications for Password-based Public Key Cryptographic Techniques", D26, September 2006. Available at `http://grouper.ieee.org/groups/1363/passwdPK/draft.html`.
9. ISO/IEC JTC 1/SC 27 11770-4, "Information Technology−Security Techniques−Key Management−Part 4: Mechanisms based on Weak Secrets", 2006. Available at `http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_tc_browse.htm?commid=45306`.
10. ITU-T Recommendation X.1035, "Password-Authenticated Key Exchange (PAK) Protocol", Series X: Data Networks, Open System Communications and Security, February 2007. Available at `http://www.itu.int/rec/T-REC-X.1035-200702-I/en`.
11. Research Papers on Password-based Cryptography. Available at `http://www.jablon.org/passwordlinks.html`.
12. J. Katz, R. Ostrovsky, and M. Yung, "Efficient and Secure Authenticated Key Exchange Using Weak Passwords", *Journal of the ACM*, Vol. 57, No. 1, pp. 78-116, 2009.
13. S. Patel, "Number Theoretic Attacks on Secure Password Schemes", In *Proc. of IEEE Symposium on Security and Privacy*, pp. 236-247, IEEE Computer Society, 1997.
14. V. Shoup, "OAEP Reconsidered", *Journal of Cryptology*, Vol. 15, No. 4, pp. 223-249, Springer-Verlag, 2002.
15. 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`.
16. S. Shin and K. Kobara, "Most Efficient Augmented Password-Only Authentication and Key Exchange for IKEv2", IETF Internet-Draft, 2010. Available at `http://tools.ietf.org/html/draft-shin-augmented-pake`.
17. T. Wu, "The SRP Authentication and Key Exchange System", IETF RFC 2945, September 2000. Available at `http://www.ietf.org/rfc/rfc2945.txt`.