Practical Authenticated Key Agreement using Passwords

Taekyoung Kwon

School of Computer Engineering, Sejong University, Seoul 143-747, Korea
E-mail: tkwon@sejong.ac.kr

Abstract. Due to the low entropy of human-memorable passwords, it is not easy to conduct password authenticated key agreement in a secure manner. Though there are many protocols achieving this goal, they may require a large amount of computation specifically in the augmented model which was contrived to resist server compromise. Our contribution in this paper is two fold. First, we propose *a new practical password authenticated key agreement protocol* that is efficient and generic in the augmented model. Our scheme is considered from the practical perspective (in terms of efficiency) and is provably secure under the Diffie-Hellman intractability assumptions in the random-oracle model. Our second contribution is more realistic and generic; *a conceptually simple but novel password guessing attack* which can be mounted on every three-pass password-based protocol unless care is taken in both the design and implementation phases. This is due to the server's failure to synchronize multiple simultaneous requests. Experimental results and possible prevention methods are also discussed.

1 Introduction

User authentication is necessary for the typical case that a human being resides as a client and tries to log on to a remote server machine. The server must be able to determine the user's identity reliably over a public or private channel. Password authentication is one of such methods, in which simply the user memorizes a (short) password while the server maintains a user profile that associates the user name and the password verifying information. The intrinsic problem with this method is the memorable password, associated with each user, has low entropy, so that it is not easy to protect the password information against the notorious password guessing attacks by which attackers could search the relatively small space of human-memorable passwords.

Since a pioneering method that resists the password guessing attacks was introduced to cryptographic protocol developers [24], there has been a great deal of work for password authenticated key agreement, preceded by EKE [5], on the framework of Diffie-Hellman [10]. Readers are referred to [15] for complete references. Compared to the typical authenticated key agreement, the password-based schemes are more expensive due to the low entropy of passwords, specifically in the augmented model which was contrived to resist server compromise. Provable security is important but tends to make the schemes harder to be practical in some cases. From the theoretical perspective, several methods that are much more expensive but provably secure in the standard model, were presented [12, 18, 19]. From the practical perspective, the practice-oriented security models are applied for examining the security of protocols [1–3, 7]. For example, EKE2 and AuthA are provably secure in both the random oracle and ideal cipher models [3, 4, 8], while PAK and PAK-Z (that improves the efficiency of PAK-X impressively by specifying a generic digital signature) are in the random oracle model [7, 25, 26]. However, it is (arguably) still expensive to assume ideal ciphers or digital signatures along with many costly operations on them, while PAK-Y is reasonably efficient with Schnorr signature in terms of computational costs [4, 26, 30].

At present, SPEKE [16], SRP [32], PAK [26], and AMP [21] are being discussed by the IEEE P1363 Standard Working Group and more recently by the ISO/IEC JTC 1/SC 27 group as practical protocols for standardization on password-based public key cryptographic techniques [13, 14]. Among them, PAK and SPEKE are 'three-pass' protocols, while AMP and SRP are 'four-pass' protocols. The standardization work is valuable in many aspects; for instance, a new attack called the 'two-for-one' guessing attack[1] against the four-pass protocols was found and resolved in the process [13, 31]. Any preference between three-pass and four-pass is still open for password-based protocols while typical authenticated key agreement such as STS and SIGMA is three-pass [11, 20].

In this paper, our contribution is two fold from the practical perspective.

1) An efficient three-pass password-based protocol in the augmented model
2) A generic password-guessing attack against three-pass protocols

A password-based protocol designed in the augmented model can resist server compromise. In other words, an adversary who compromised a password profile from a server cannot impersonate a user without launching dictionary attacks. For this additional property, the related protocols (for example, A-EKE, AMP, AuthA, B-SPEKE, PAK-Z, and SRP) are more expensive than those are not (for instance, EKE, EKE-2, SPEKE, and PAK) in the augmented model [6, 21, 4, 17, 26, 32]. We observe that the existing provably-secure schemes are still expensive in the augmented model in terms of the amount of computation, and that it is desirable to minimize the number of message passes and the size of message blocks for practice on expensive communication channels. So we design a new three-pass password-based protocol in the augmented model with both security and efficiency in mind. We achieve this goal interestingly by a composition under the careful observation of the existing schemes discussed by the IEEE P1363 Standard Working Group, say without losing the presumed level of security. We call the protocol TP-AMP and prove its security in the random oracle model.

On developing the new three-pass password-based protocol, we find a conceptually simple but novel password guessing attack which can be mounted on every

[1] An active attacker can validate two password guesses in one impersonation attempt. The first attack against SRP was discovered by D. Bleichenbacher in 2000, while the similar attack on AMP was by M. Scott [31]. However, both protocols were fixed to resist respective attacks by each original author [13].

three-pass password-based protocol by exploiting a small window of vulnerability resulting from a standard technique to resist on-line guessing attacks, say from counting the number of failed requests. Our attack is due to the server's failure to synchronize multiple simultaneous requests, and is unavoidable in three-pass protocols unless special care is taken in both the design and implementation phases. We call this attack a *many-to-many* (or *parallel*) guessing attack[2] because an active attacker can validate as many password guesses as (s)he makes server instances invoked concurrently, regardless of its upper limit of on-line guessing. A prototype of the proposed protocol is implemented to show how our attack works and is prevented. We first consider this attack and possible resolution in the literature.

This paper is organized as follows. In the following section, the so-called TP-AMP protocol (our first contribution) is presented. In Section 3, the many-to-many guessing attack (our second contribution) is described in more detail. In Section 4, security and efficiency of TP-AMP are discussed. Finally this paper is concluded in Section 5.

2 A Practical Protocol

2.1 Preliminaries

Our principal motivation comes from the fact that password-based protocols designed in the augmented model are much less efficient than those are not in that model, in terms of either computation or communication costs. When we regard PAK as a fundamental structure for three-pass protocols due to its simplicity and clarity, we can easily observe that its augmentation such as PAK-X, PAK-Y, and PAK-Z are far from its intrinsic nature and get much more complicated in the augmented model [7, 25, 26]. AMP and SRP show better performance in that model but in four passes [21, 32]. So, our basic idea is to make AMP squeezed into PAK or PAK augmented by AMP, since AMP is another protocol that can be computed very efficiently over various numerical groups [21]. However, a simple composition is not sufficient, and consequently we obtain a new practical protocol by more careful consideration on them.

The reason for choosing PAK rather than EKE2 is obviously that the former can formally be proved by postulating the random oracles only, while the latter requires the additional assumption of ideal cipher [7, 26, 3]. However, EKE2 or similar schemes that are proved sufficiently secure, can also be applied to constructing the practical augmented protocol in the way of our composition. In that sense, our construction is quite generic.

In Table 1, we enumerate the notation, in part, to be used in the remaining of this paper. Additional ones will be self-contained in each part of this paper. Let κ be a general security parameter (say 160 bits) and ℓ be a special security parameter for public keys (1024 or 2048 bits). A client C and a server S should

[2] We first introduced this attack at IEEE P1363.2 meeting and also discussed a few names for it.

Table 1. Basic Notation

C	Client (User)	S	Server
π	Password	τ_C	Transformed password for C
$\leftarrow$	Derivation	$\overset{\mathrm{R}}{\leftarrow}$	Random selection
κ, ℓ	Security parameters	q	Prime of size κ
r	Integer co-prime to q	p	Prime of size ℓ such that $p = rq + 1$
$\mathbb{Z}_p^*$	Multiplicative group of p	$\mathbb{G}_q$	q-order subgroup of $\mathbb{Z}_p^*$
g	Generator of $\mathbb{G}_q$	h_i, H_i	Random oracles
α, β	Agreed values	sk_i	Session key

agree on algebraic parameters[3] related to Diffie-Hellman key agreement such as p, q, and g. Define $\bar{\mathbb{G}}_q = \{g^x \bmod p | x \in \mathbb{Z}_q^*\}$ where $|\bar{\mathbb{G}}_q| = q-1$. Let us often omit 'mod p' from the expressions that are obvious in $\mathbb{Z}_p^*$. Let $\{0,1\}^*$ denote the set of finite binary strings and $\{0,1\}^n$ the set of binary strings of length n. We then define random oracles such that h_i: $\{0,1\}^* \rightarrow \{0,1\}^\kappa$ and H_i: $\{0,1\}^* \rightarrow \{0,1\}^\ell$. Their instances are also defined as $h_i(\cdot) = h(i, \cdot, i)$ and $H_i(\cdot) = (h(i, \cdot, i))^{\frac{p-1}{q}}$ mod p where $h(\cdot)$ is a strong one-way hash function. Let $\textsf{ACCEPTABLE}(\cdot)$ denote an acceptable function which may return true if its pre-image satisfies the given security properties, as defined in Section 2.3. Readers who are not familiar with the legacy protocols, are referred to the previous work of [7, 13, 21, 25, 26].

2.2 Proposed Protocol - TP-AMP

TP-AMP stands for the Three-Pass Authenticated key agreement via Memorable Passwords and is depicted in Figure 1. Let us borrow the name AMP from [21] for our basic motivation.

Protocol Setup On the registration phase, a user chooses a name C and a password π while the server S saves user's profile $\langle C, \tau_C \rangle$ in its stable storage where $\gamma = H_0(C, \pi)$, $\gamma' = \gamma^{-1} \mod p$, $u = h_1(C, \pi)$, $\nu = g^u$, and $\tau_C = \langle \gamma', \nu \rangle$. For convenience, S is assumed as an IP address of the server machine.

Protocol Run A user may type C and π into the client machine. The client (C on behalf of the user from now on) then chooses x at random from $\mathbb{Z}_q^*$ (not $\mathbb{Z}_p^*$), and computes γ in order to obtain $m = g^x\gamma$. The client sends ($\rightarrow$) a commitment message $\langle C, m \rangle$ to the server.

$$1.\ C \rightarrow S : C, g^x\gamma$$

[3] In spite that PAK, in general, does not require gcd(r, q)=1 and only PAK-R requires it for further randomization, we recommend to use a *secure prime* such that each factor of r except 2 is of size at least κ or a *safe prime* such that $r = 2$ for $p = rq+1$ as discussed in [21, 23, 32, 29]. They satisfy gcd(r, q)=1. Specifically, we observe that TP-AMP shows the best performance with a secure prime, while PAK-Y does with a safe prime and arbitrarily smaller exponents [28].

$$
\begin{array}{lcl}
\mathsf{client}[C,\pi] & & \mathsf{server}[C,\tau_C(\gamma',\nu)] \text{ on } S \\
 & & \langle \gamma' = \gamma^{-1} \bmod p, \nu = g^u \bmod p \rangle \\
x \overset{\text{R}}{\leftarrow} \mathbb{Z}_q^* & & \\
\gamma \leftarrow H_0(C,\pi) & & \\
m \leftarrow g^x\gamma \bmod p & & \\
 & \xrightarrow{C,m} & \\
 & & \text{abort if } \neg\mathsf{ACCEPTABLE}(C,m) \\
 & & y \overset{\text{R}}{\leftarrow} \mathbb{Z}_q^* \\
\gamma' \leftarrow \gamma^{-1} \bmod p & & \mu \leftarrow \nu^y \bmod p \\
u \leftarrow h_1(C,\pi) & & \beta \leftarrow (m\gamma' g^m)^y \bmod p \\
w \leftarrow u^{-1}(x+m) \bmod q & & k_1 \leftarrow h_2(C,S,m,\mu,\beta,\gamma') \\
 & \xleftarrow{\mu, k_1} & \\
\alpha \leftarrow \mu^w \bmod p & & \\
k_1' \leftarrow h_2(C,S,m,\mu,\alpha,\gamma') & & \\
\text{abort if } k_1 \neq k_1' & & \\
k_2 \leftarrow h_3(C,S,m,\mu,\alpha,\gamma') & & k_2' \leftarrow h_3(C,S,m,\mu,\beta,\gamma') \\
 & \xrightarrow{k_2} & \\
 & & \text{abort if } k_2 \neq k_2' \text{ or time is out} \\
sk_C \leftarrow h_4(C,S,m,\mu,\alpha,\gamma') & & sk_S \leftarrow h_4(C,S,m,\mu,\beta,\gamma')
\end{array}
$$

Fig. 1. TP-AMP (Three-Pass AMP Protocol)

After or before sending message 1, the client could compute γ' and the user's *amplified password* such that $w = u^{-1}(x+m) \bmod q$ by obtaining $u = h_1(C,\pi)$, and keeps them while waiting for message 2. In practice, we can hash m so that we have $q|h(m)$ with negligible probability.

Upon receiving message 1, the server should abort it if ACCEPTABLE (C,m) returns false. Otherwise, the server fetches $\langle C, \tau_C \rangle$ from its storage and chooses y at random from $\mathbb{Z}_q^*$ so as to obtain $\mu = \nu^y$. The server then computes $\beta \equiv (m\gamma' g^m)^y \equiv g^{(x+m)y} (\text{mod } p)$ and $k_1 = h_2(C,S,m,\mu,\beta,\gamma')$, and sends a challenge message $\langle \mu, k_1 \rangle$ to the client.

$$2.\ S \to C : \nu^y, h_2(C,S,m,\mu,\beta,\gamma')$$

After or before sending message 2, the server could compute $k_2' \leftarrow h_3(C,S,m,\mu,\beta,\gamma')$ and keeps it while waiting for message 3. The server should abort if time is run out.

Upon receiving message 2, the client raises μ to the amplified password so that $\alpha \equiv \mu^w \equiv g^{y(x+m)} (\text{mod } p)$, and computes $k_1' = h_2(C,S,m,\mu,\alpha,\gamma')$. If k_1 is not equal to k_1', the client should abort this session. Otherwise, the client computes $k_2 = h_3(C,S,m,\mu,\beta,\gamma')$ and sends a response message k_2 to the server.

$$3.\ C \to S : h_3(C,S,m,\mu,\alpha,\gamma')$$

After or before sending message 3, the client could compute a session key such that $sk_C = h_4(C,S,m,\mu,\alpha,\gamma')$ and deletes any other ephemeral values.

Upon receiving message 3, the server should abort this session if k_2 is not equal to k'_2. Otherwise, the server should compute a session key such that $sk_S = h_4(C, S, m, \mu, \beta, \gamma')$ and deletes any other ephemeral values.

As a result, the client and the server could authenticate each other using the passwords and agree on the same session key $sk_C(= sk_S)$ because $\alpha \equiv \beta \equiv g^{(x+m)y} (\bmod p)$.

2.3 Small Discussion

One can easily see that message 1 is extracted from PAK while message 2 and session key are motivated by AMP. This protocol performs simple computation in three passes and works in the augmented model where τ_C is defined as $\langle \gamma', \nu \rangle$. For efficiency, it would be better to hash m when we compute β and w, say $\beta = (m\gamma' g^{h(m)})^y$ and $w = u^{-1}(x + h(m)) \bmod q$ for a strong one-way hash function $h(\cdot)$. For more efficiency, we recommend to use a secure prime for TP-AMP. Security and efficiency of the proposed protocol will be discussed in Section 4.

In the legitimate protocol run, g^x and ν^y are assumed not to be trivial values such as 0 and 1 as in the Diffie-Hellman relatives. We need to define a *failure count* that must be manipulated by the server and increased by one when $k_2 \neq k'_2$. The server should abort further requests of the client if the (subsequent) failure count exceeds its pre-defined limit, δ. This is a standard technique for resisting on-line guessing attacks. We also need to define the special function called ACCEPTABLE$(\cdot)$ since the server should abort when it returns false upon receiving $\langle C, m \rangle$. An example of the function follows:

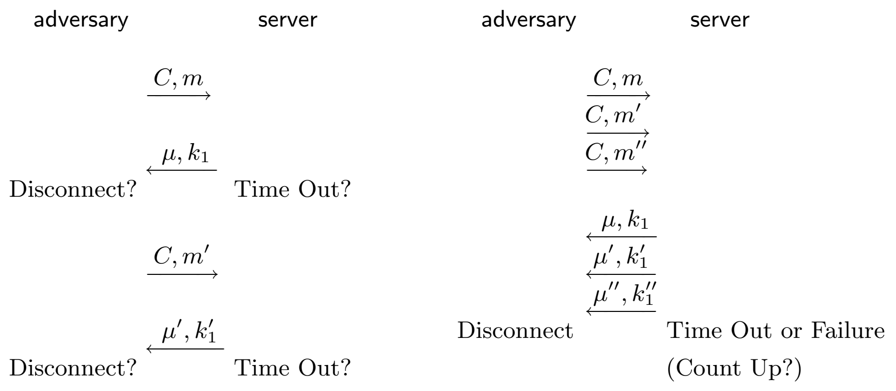
```
ACCEPTABLE(·)
INPUT: $\langle C, m \rangle$
OUTPUT:
Return *false*
  if $C$ is being served by another instance; /* See Section 3 */
  else if the failure count of $C$ is greater than or equal to its limit $\delta$;
  else if $q|m$; /* Check if $m \notin Z_p^*$ only when hashing $m$ before raising $g$ */
Return *true* otherwise;
```

Note that the first condition (for resisting the many-to-many guessing attacks in the next section) can be considered in very flexible ways, for example, an IP address instead of C, and can be substituted by a more effective way in the future. This function is valid for authentication sessions only. Note also that $q|m$ means q divides m, but it might be enough to assure $m \in Z_p^*$ only when we hash m for β and w in the protocol.

3 Many-to-Many Guessing

3.1 A Real World Attack

It is widely recognized that three-pass (say, smaller-pass) protocols are favorable to the channel efficiency for authenticated key agreement. However, care must be taken for password authenticated key agreement in a practical sense.



(a) Sequential Attempts
- Exercise -

(b) Parallel Attempts
- Real Attack -

Fig. 2. Basic Concept of Many-to-Many Guessing Attacks

Let us glance over Theorem 1, in advance, that is introduced in Section 4 and proved in [22]. There exists an adversarial advantage that is bounded by $\frac{q_{se}}{N}$. The similar results can be found from the closely related work [3, 8, 26]. These advantages imply that the adversary is reduced to a simple online guessing attacker that can easily be detected and prevented from exceeding the pre-defined limit, δ, on the number of sequential on-line trials allowed by the server's policy. For example, an adversary posing as a user C sends an arbitrary message $\langle C, m\rangle$ to the server, based on her guessed password. The server may respond with $\langle \mu, k_1\rangle$ in the three-pass protocols while only μ in the four-pass protocols. Then, the adversary is assumed to check her guess with probability bounded by $\frac{q_{se}}{N}$ under the limit δ in three-pass protocols. Is this standard assumption really true?

Unfortunately, the answer is No! This classical prevention method can be fooled out of making the adversarial advantage much larger and in some cases disclosing a password, in a surprisingly simple way. Figure 2 depicts the possible bad events. Our attack is motivated from the fact that the server is typically implemented as a multi-threaded or multi-process application for handling many user requests simultaneously, and that the three-pass password-based protocol is not an exception. As summarized in Figure 2-(a), the adversary is able to exercise the real attack (that is described in Figure 2-(b)), for example, in order to approximate the maximum amount of time the server may wait for the third message k_2. The adversary then starts simultaneous authentication sessions, which the server processes independently in separate threads, and in that amount of time, is able to drive many different initiating messages based on different password guesses concurrently to the server. The adversary may get as many replies as allowed in that time boundary, by exceeding δ obviously. Figure 2-(b) abbreviates this idea. It could be a real world attack from the automated (and multi-threaded) adversary. The server instances must respond to each request and wait for the replies k_2 from the adversary who can even disconnect without answering, for example, by manually unplugging the network cable or automatically manipulating the transport layer.

As a result, the adversary is able to gather many triples, $\langle m, \mu, k_1 \rangle$, and mount the further guessing attacks off-line. The adversary is able to check *many* password guesses over δ while the server may notice *many* guessing attempts bounded by δ afterwards. So we call this simple attack the *many-to-many guessing attack*[4]. The window of vulnerability can be thought of as

$$O(T) = t_S + t_C + 2t_{CS} + \varepsilon + (\delta - 1)\epsilon$$

where 1) t_S is the time between receiving $\langle C, m \rangle$ and sending out $\langle \mu, k_1 \rangle$ in the server, 2) t_C is the time between receiving $\langle \mu, k_1 \rangle$ and sending out k_2 in the client, 3) t_{CS} is the time delay for exchanging messages over a communication channel, 4) ε is the (most influential) additional waiting time defined by the server considering T_C and T_{CS}, and 5) ϵ is the (most negligible) amount of time that defines the average time difference between one request and another subsequent one (so, $(\delta - 1)\varepsilon$ must be the time between the first notice of a failed attempt and the last one under δ). We address that this window of vulnerability is not negligible and is sufficient to allow the adversary to gather as many triples as she can fool the protocol out of being over δ and disclosing the password in the worst case.

3.2 Possible Prevention

We designed the ACCEPTABLE$(\cdot)$ function in Section 2.3 so as to return false if C (or a corresponding IP address) is being served already by another server instance upon receiving a new message $\langle C, m \rangle$. Note that the 'serving' corresponds to the authentication session only. This was actually contrived for resisting the many-to-many guessing attack. For the purpose, a small hash table may be maintained by the server to track the currently served or blocked clients. The blocking policy should be considered carefully but flexibly. This resolution method may reduce the window of vulnerability notably but still leaves an issue about DoS (Denial of Service). Note that there is a recent literature considering DoS attacks on password-based protocols [9]. Aside from the danger of DoS attacks, a race condition and some bottleneck to the hash table are now only concerns while they could be negligible by careful consideration. The possible prevention methods might be considered both in the design and implementation phases.

In order to examine the reality of our attack, we implement a prototype of TP-AMP and launch the many-to-many guessing attack on it. We implement

[4] This attack is negligible in the four-pass protocols since the server does not give sufficient information to the adversary forward and the client is usually not capable of listening to so many concurrent requests in the opposite case. Also, the best-known predecessors, EKE [5] and A-EKE [6], avoid this attack very *impressively* by not optimizing the protocol steps.

both client and server using `CreateThread(·)` functions and `WinSock` in Pentium IV 1.8GHz, 512MB, MS-Windows platforms. Simply a single run takes 297 milliseconds. We then drive multi-threaded clients to starts many simultaneous authentication sessions. We summarize the results. Let $\delta = 5$ (times) and $\varepsilon = 3$ (seconds), without correct ACCEPTABLE$(\cdot)$ function. Until we increase the number of adversarial client threads to 100, we could observe all requests are stably served and the same number of triples, $\langle m, \mu, k_1 \rangle$, are gathered. When 150 threads are driven, about 20 requests are only declined, and about 130 triples are gathered. However, about 145 threads are blocked with correct ACCEPTABLE$(\cdot)$ function.

4 Security and Efficiency Analysis

In this section, we discuss security and efficiency of TP-AMP.

4.1 Security of TP-AMP

For formal security, we adapt the improved models of [26] and [8]. Our refreshed security model is described in [22]. Readers are referred to it due to the page restriction of this paper. We prove that the TP-AMP protocol is secure, in the sense that an adversary attacking the system cannot determine session keys of fresh instances with greater advantage than that of an online dictionary attack, and cannot determine session keys of semi-fresh instances with greater advantage than that of an off-line dictionary attack. We define q_{se}, q_{ex}, q_{re}, q_{co} queries as those of type Send, Execute, Reveal, Corrupt, respectively, and q_{ro} queries to be made to the random oracles. Also we define some events related to the adversary making a password guess. Note that $[\alpha, \beta]$ means one of α and β is drawn. Also recall that the order of $\bar{\mathbb{G}}_q$ is $q-1$. Security arguments for the following theorems are described in [22].

Theorem 1. *Let $\mathcal{P}$ be the TP-AMP protocol with a password dictionary of size N. Fix an adversary $\mathcal{A}$ that runs in time t, and makes q_{se}, q_{ex}, q_{re}, q_{co} queries and q_{ro} queries. Then for $t' = O(t + (q_{ro} + q_{se} + q_{ex})t_{\mathsf{exp}})$ with t_{exp} denoting the computational time for exponentiation in $\mathbb{G}_q$:*

$$\mathsf{Adv}_{\mathcal{P}}^{\mathsf{ake-fs}}(\mathcal{A}) \leq \frac{q_{se}}{N} + O((q_{se}+q_{ex})q_{ro}\mathsf{Adv}_{\mathbb{G}_q}^{\mathsf{CDH}}(t', q_{ro})) + \frac{O((q_{se}+q_{ex})^2)}{q-1} + \frac{O(q_{se}+q_{ro}^2)}{2^{\kappa}}$$

and

$$\mathsf{Adv}_{\mathcal{P}}^{\mathsf{ake-fs.s}}(\mathcal{A}) = \frac{q_{ro}}{N} + \mathsf{Adv}_{\mathcal{P}}^{\mathsf{ake-fs}}(\mathcal{A}).$$

Proof sketch: Our proof will proceed by defining a sequence of games starting at the real game $\mathbf{G}_0$ and ending up at $\mathbf{G}_7$. In the beginning we simulate all protocol queries, and remove possible collisions and lucky events. We then reduce our protocol from solving CDH in a stringent way, for respective Execute and Send queries. So $\mathbf{G}_5$ models passive adversaries, while $\mathbf{G}_6$ does active adversaries. Finally, server compromise is manipulated in $\mathbf{G}_7$. □

Theorem 2. *Let $\mathcal{P}$ be the TP-AMP protocol with a password dictionary of size N. Fix an adversary $\mathcal{A}$ that runs in time t, and makes q_{se}, q_{ex}, q_{re}, q_{co} queries and q_{ro} queries. Then for $t' = O(t + (q_{ro} + q_{se} + q_{ex})t_{\mathsf{exp}})$ with t_{exp} denoting the computational time for exponentiation in $\mathbb{G}_q$:*

$$\mathsf{Adv}_{\mathcal{P}}^{\mathsf{ma}}(\mathcal{A}) \leq \frac{q_{se}}{N} + O((q_{se}+q_{ex})q_{ro}\mathsf{Adv}_{\mathbb{G}_q}^{\mathsf{CDH}}(t', q_{ro})) + \frac{O((q_{se} + q_{ex})^2)}{q-1} + \frac{O(q_{se} + q_{ro}^2)}{2^{\kappa}}$$

and

$$\mathsf{Adv}_{\mathcal{P}}^{\mathsf{c2s.s}}(\mathcal{A}) = \frac{q_{ro}}{N} + \mathsf{Adv}_{\mathcal{P}}^{\mathsf{ake-fs}}(\mathcal{A}).$$

We believe the given security argument in the random oracle model in [22] is sufficient to ensure that TP-AMP is a secure password authenticated key agreement protocol in the augmented model, though a full proof might be more intricate. A resistance to our real world attack can be observed by manipulating the ACCEPTABLE$(\cdot)$ function. Since TP-AMP is simple in its structure, it might also be easy and obvious to examine its security heuristically but we do not manipulate any heuristic analysis in this paper.

4.2 Efficiency of TP-AMP

We may consider the number of expensive operations, for example, multiple precision multiplications (MPM) in $\mathbb{Z}_p^*$, in order to analyze the performance of TP-AMP. We approximate the number of multiplications on average, by assuming the use of a left-to-right binary exponentiation method or a simultaneous exponentiation method (denoted by sim) [27]. A slight computational difference between squaring and multiplication can be ignored for convenience.

Let $\kappa' = \kappa'' = \kappa''' = \kappa$ for secure prime p while $\kappa' = \kappa'' = \kappa''' = \ell - 2/3$ for safe prime p for simple analysis. Practically we can set $\kappa'' = 2\kappa$ with intentionally smaller exponents, for example, $x \overset{\text{R}}{\leftarrow} \{0,1\}^{\kappa''}$ and not $\overset{\text{R}}{\leftarrow} \mathbb{Z}_q^*$ for safe prime p though it is not general for security argument. For the client, γ may take $1.5(\ell - \kappa')$ while m may need $1.5\kappa'' + 1$ MPM. For the server, β may take $1.5(\kappa + \kappa'') + 2$ or $2\kappa'' + 1(\mathsf{sim})$ MPM. Finally the client may need $1.5\kappa'''$ for α. As a result, the client may need $1.5(\ell - \kappa' + \kappa'' + \kappa''') + 1$ while the server may require $1.5(\kappa + 2\kappa'') + 2$ MPM totally. Also if we assume an ideal cipher for γ as like EKE2 and AuthA [3, 4], the client may need $1.5(\kappa'' + \kappa''')$ MPM only. One can easily see that the TP-AMP protocol is efficient especially when we use a secure prime, p, since $\kappa' = \kappa'' = \kappa''' = \kappa$.

TP-AMP is comparable to the most closely related protocol PAK-Z (with an efficient instance Y using Schnorr's signature [30]) [25, 26] and AuthA [4, 8] in terms of efficiency. In general, TP-AMP is more efficient than those related schemes under the same assumption, for example, on a safe or secure prime with group size exponents, or an ideal cipher. However, when we use a safe prime with intentionally smaller exponents, PAK-Y shows better in the client, while TP-AMP does still better in the server. Though TP-AMP is not a direct instance of PAK-Z, we can say it provides efficiency on the framework of PAK in the augmented model. As we mentioned already, TP-AMP can be instantiated on EKE2 [3, 4, 8] and provide efficiency on that framework. Thus, we would like to address that TP-AMP is a practical password authenticated key agreement protocol with sufficient security and generic features in the augmented model.

5 Conclusion

Though three-pass protocols may reduce one-round from four-pass protocols and are easier to apply provable security, we show that they are vulnerable to the novel many-to-many password guessing attacks for password authenticated key agreement. From the practical perspective we design and analyze a new three-pass protocol, TP-AMP, in the augmented model and show several interesting features. In the future study, we will conduct more intensive work on three-pass and four-pass protocols for password authenticated key agreement.

Acknowledgement

This work was supported in part by Korea Research Foundation Grant (KRF-2003-003-D00434). The author thanks anonymous referees for kind comments.

References

1. M. Bellare and P. Rogaway, "Entity authentication and key distribution," In *Crypto 1993*, LNCS 773, pp.232-249, 1993.
2. M. Bellare and P. Rogaway, "Provably secure session key distribution-the three party case," In *ACM Symposium on the Theory of Computing*, pp.232-249, 1993.
3. M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attack," In *Eurocrypt 2000*, LNCS 1807, pp.139-155, 2000.
4. M. Bellare and P. Rogaway, "The AuthA protocol for password-based authenticated key exchange," Submission to the IEEE P1363.2 study group, available from `http://www.cs.ucdavis.edu/~rogaway /papers/autha.ps`
5. S. Bellovin and M. Merritt, "Encrypted key exchange : password-based protocols secure against dictionary attacks," In *IEEE Symposium on Research in Security and Privacy*, pp. 72-84, 1992.
6. S. Bellovin and M. Merritt, "Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password-file compromise," In *ACM Conference on Computer and Communications Security*, pp. 244-250, 1993.
7. V. Boyko, P. MacKenzie and S. Patel, "Provably secure password authenticated key exchange using Diffie-Hellman," In *Eurocrypt 2000*, LNCS 1807, pp.156-171, 2000.
8. E. Bresson, O. Chevassut, and D. Pointcheval, "Security proofs for an efficient password-based key exchange," In *ACM Conference on Computer Communications Security*, 2003.
9. E. Bresson, O. Chevassut, and D. Pointcheval, "New security results on Encrypted Key Exchange," In *International Workshop on Theory and Practice in Public Key Cryptography*, LNCS 2947, pp. 145-158, 2004.

10. W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol.22, no.6, pp.644-654, November 1976.
11. W. Diffie, P. van Oorschot, and M. Wiener, “Authentication and authenticated key exchanges,” Designs, Codes and Cryptography, 2, pp. 107-125, 1992.
12. O. Goldreich and Y. Lindell, “Session-Key Generation Using Human Passwords Only,” In *Cypto 2001*, LNCS 2139, pp.408-432, 2001.
13. IEEE P1363.2, *Standard specifications for password-based public key cryptographic techniques*, available from `http://grouper.ieee.org/groups/1363/`, December 2002.
14. ISO/IEC WD 11770-4, *Information technology - Security techniques - Key management - Part 4: Mechanisms based on weak secrets*, ISO/IEC JTC 1/SC 27, November 2003.
15. Phoenix Technologies, Inc., “Research Papers on Strong Password Authentication,” available from `http://www.integritysciences.com/links.html`, 2002.
16. D. Jablon, “Strong password-only authenticated key exchange,” *ACM Computer Communications Review*, vol.26, no.5, pp.5-26, 1996.
17. D. Jablon, “Extended password key exchange protocols,” In *WETICE Workshop on Enterprise Security*, pp.248-255, 1997.
18. J. Katz, R. Ostrovsky, and M. Yung, “Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords ,” In *Eurocrypt 2001*, LNCS 2045, pp.475-494, 2001.
19. K. Kobara and H. Imai, “Pretty-simple password-authenticated key-exchange protocol proven to be secure in the standard model,” IEICE Trans., E85-A(10), pp.2229-2237, 2002.
20. H. Krawczyk, “SIGMA: The ‘SINGn-and-MAc’ approach to authenticated Diffie-Hellman and its use in the IKE protocols,” *Advances in Cryptology - CRYPTO 2003*, Lecture Notes in Computer Science, Vol. 2729, Springer-Verlag, pp. 400-425, 2003.
21. T. Kwon, “Authentication and key agreement via memorable password,” In *ISOC Network and Distributed System Security Symposium*, February 2001.
22. T. Kwon, “Practical authenticated key agreement using passwords,” Full version of this paper, available from `http://dasan.sejong.ac.kr/∼tkwon/amp.html`.
23. C. Lim and P. Lee, “A key recovery attack on discrete log-based schemes using a prime order subgroup,” In *CRYPTO 97*, pp.249-263, 1997.
24. M. Lomas, L. Gong, J. Saltzer, and R. Needham, “Reducing risks from poorly chosen keys,” In *ACM Symposium on Operating System Principles*, pp.14-18, 1989.
25. P. MacKenzie, “More efficient password-authenticated key exchange,” In *RSA Conference*, Cryptographers Track, LNCS 2020, pp.361-377, 2001.
26. P. MacKenzie, “The PAK suite: Protocols for Password-Authenticated Key Exchange,” Submission to IEEE P1363.2, April 2002.
27. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of applied cryptography*, CRC Press,Inc., pp.517-518, 1997.
28. P. van Oorschot and M. Wiener, “On Diffie-Hellman key agreement with short exponents,” In *Eurocrypt 96*, pp. 332-343, 1996.
29. R. Perlman and C. Kaufman, “PDM: A new strong password-based protocol,” In *USENIX Security Symposium*, pp.313-321, 2001.
30. C. Schnorr, “Efficient identification and signatures for smart cards,” In *Crypto 89*, pp.239-251, 1989.
31. M. Scott, *Personal communication*, July 2001.
32. T. Wu, “Secure remote password protocol,” In *ISOC Network and Distributed System Security Symposium*, 1998.