Ultimate Solution to Authentication via Memorable Password

Taekyoung Kwon (*tkwon@cs.berkeley.edu*) - May 2000 version

Abstract

Human-memorable password authentication is not easy to provide over insecure networks due to the low entropy of the password. Such a password is typically vulnerable to dictionary attacks. A cryptographic protocol is the most promising solution to this problem. So far, numerous password authentication protocols have been proposed. Among them, A-EKE is a great landmark of verifier-based protocol and is followed by many distinguished protocols[7, 18, 34, 23, 9] such as SRP that is notable in its efficiency and SNAPI-X that is the first provable approach of those protocols[34, 23]. Verifier-based protocols allow the asymmetric model in which a client possesses a password, while a server stores its verifier. Inspired by those works, this paper introduces a new protocol called AMP in a provable manner. It is the ultimate result of the author's AMP(Authentication and key agreement via Memorable Password) research project. AMP allows the Diffie-Hellman based key agreement and is actually superior to other related work in terms of efficiency and generalization features. We give a rigorous comparison to them.

1 Introduction

Entity authentication is one of the most important security functions. It is necessary for verifying the identities of the communicating parties when they initiate a connection. This function is usually provided in combination with a key establishment scheme such as key transport or key agreement between the parties. For user authentication, three kinds of approaches exist which may be combined for sophisticated use; knowledge-based authentication, token-based authentication, and biometric authentication. Among them, the knowledge-based scheme needs knowledge-proofs so that it is only for human mind ($\approx$ memory). Actually, it is the most widely-used method due to the advantages of simplicity, convenience, adaptability, mobility, and less hardware requirement. It requires users only to remember and type in their knowledge such as a password or PIN(personal identification number). Therefore, users are allowed to move conveniently without carrying hardware tokens; the user knowledge is also useful for unlocking such hardware tokens. However, a complex problem with this password-only authentication is that a human-memorable password having low entropy allows malicious guessing attacks. The problem becomes much more critical in an open distributed environment. A cryptographic protocol is the most promising solution to it.

Password Protocols.[1] Since the first scheme, LGSN[21], was introduced in 1989, many protocols have followed it. Among them, EKE[6] was a landmark of certificate-free protocols though it gave several strong constraints due to the symmetric encryption of a public-key. One variant of EKE, DH-EKE[6], introduced the password authentication and key agreement, and was "enhanced" to A-EKE[7] that was the first verifier-based protocol to resist a password-file compromise and to accommodate salt. GLNS[14] is enhanced from LGSN. Due to the inefficiency and constraints of older schemes, several modifications and advanced descendents were spawned[33, 1, 32, 15, 19, 17, 18, 30, 34, 16]. However, some of them have been broken and some are still being cryptanalyzed[2, 12, 27, 8]; most were inadequate for the security proof due to ad-hoc methods of protecting the password. In the mean time, OKE was introduced as the first provable approach based on the work of Bellare and Rogaway[22, 3] and was followed by SNAPI[23]. Halevi and Krawczyk's work also intended a provable approach but their protocol requires users to keep some additional information called a public password[16]. Most recently, AuthA and PAK have been introduced separately[5, 9] and they show the provable approach in this area is getting matured[4].

A family of the password-verifier based protocol is composed of A-EKE, B-SPEKE, SRP, GXY, SNAPI-X, AuthA, and PAK-X[7, 18, 34, 20, 23, 5, 9]. The verifier-based protocols allow the asymmetric model in which a client possesses a password, while a server stores its verifier rather than the password itself. A-EKE was the first verifier-based version augmented from DH-EKE[7]. B-SPEKE was augmented from SPEKE which was efficient without a verifier[17, 18]. SRP was efficient and practical even with a verifier[34]. GXY was derived from SRP for agreeing on the Diffie-Hellman exponential[20]. SNAPI-X was augmented from SNAPI and it was fully provable in the random oracle model[23]. AuthA was newly proposed but very similar to the previous protocols[5]. PAK-X was enhanced from PAK[9] and it was so recent version of those protocols that we could find it when we were on finishing this paper. We will give a rigorous comparison of our protocol, AMP, to those protocols in section 4.3.

This Paper. Our goal is to design the password-verifier based authentication protocol in a provable manner, which allows a server-compromise resistance via verifiers and a secure key agreement by the Diffie-Hellman scheme. The protocol must be efficient and easy to generalize by utilizing the main group operation. Our protocol is actually called u-AMP(Ultimate Authentication and key agreement via Memorable Password) because it is an ultimate result of AMP[2] research project. However, we call it simply AMP in this paper. Actually, it is the most efficient protocol among the existing verifier-based protocols. This paper is structured as follows. The remaining part of this section summarizes a notation partly. Section 2 shows

[1] Readers are referred to Figure 3 attached in Appendix of this paper. Jablon's work[17] is recommended as the best tutorial for the password protocol study while Wu's work[34] is the best for the verifier protocol study. Bellare and Rogaway's work[3] is the fundamental of the provable approach.

[2] AMP implies a password amplifier which means a method of strengthening the low-entropy password.

our protocol design concept formally, including our basic idea and proof model. Section 3 proposes our protocol and examines the security in the random oracle model. Section 4 analyzes various features of our protocol and gives a rigorous comparison to other related protocols. Section 5 concludes this paper.

NOTATION. A notation that is not introduced here will be defined in each part of this paper. Since our protocol is typically the two party case, we use *Alice* and *Bob* for describing a client and a server, respectively. *Eve* indicates an adversary whether she is passive or active. π and τ denotes a password and salt, respectively. Let g be a generator of a large prime-order subgroup Z_q^* or a multiplicative group $Z_p^*$where $p = qr + 1$; q is a large prime factor of huge prime p while r is a composite of prime integers. Note that we abbreviate a modular notation, mod p, for convenience hereafter. $\doteq$ means a comparison of two terms, for example, $\alpha \doteq \beta$. Let $\{0,1\}^*$ denote the set of finite binary strings and $\{0,1\}^\infty$ the set of infinite ones. λ implies the empty string. k is our security parameter long enough to prevent brute-forcing while $l(k) \geq 2k$, $\omega(k) \leq \frac{2}{3}k$, and $t(k) \leq \frac{1}{3}k$. $h() : \{0,1\}^* \rightarrow \{0,1\}^{l(k)}$ means a collision-free one-way hash function. $\varphi() : \{0,1\}^* \rightarrow \{0,1\}^{\leq 2l(k)}$ means a one-way function. All hash functions are assumed to behave like random oracles for security proof[3].

2 Protocol Design Concept

Our goal is to design a protocol which combines the following functions securely and efficiently.

- Password-verifier based authentication[7]
- Diffie-Hellman key agreement[11]
- Easy generalization [24]

This section describes our protocol design concept and security model.

2.1 A Conceptual Design

The security of AMP is based on the following well-known hard problems which are believed infeasible to solve in polynomial time.

Discrete Logarithm Problem Given a prime p, a generator g of a multiplicative group Z_p^*, and an element $g^x \in Z_p^*$, find the integer $x \in [0, p-2]$.

Diffie-Hellman Problem Given a prime p, a generator g of a multiplicative group Z_p^*, and elements $g^x \in Z_p^*$ and $g^y \in Z_p^*$, find $g^{xy} \in Z_p^*$.

These two problems hold their property also in a large prime-order subgroup[26].

The Verifier. A verifier $\mathcal{V}$ is the information computed from salt τ and password π. It is composed of an explicit verifier ν retained by a server and an implicit verifier υ obtained by a client. ν is computable from υ whereas the reverse is infeasible in polynomial time.

Definition 1 *Our password-verifiers are defined by,*

- explicit verifier : $\nu = g^{\upsilon}$,
- implicit verifier : $\upsilon = \varphi(\tau, \pi)$.

Two verifiers are used in the way that a compromised password-file shows the explicit verifier whereas the implicit verifier is actually necessary for authentication. $\varphi()$ is a one-way function that merely perturbs and expands τ and π into a secure exponent[26], $\varphi() \in \{0,1\}^{\leq 2l(k)}$.

The Amplification and Key Exchange. The amplification of the password entropy means *Alice* never shows the password itself but rather hides it by merging with the high entropy information such as the Diffie-Hellman exponentials when she proves knowing it. If one party commits the high entropy information to the other party and receives the high entropy challenge, then he or she can reply with the amplified password information. The amplified password information only shows the fact that the party knows the password, rather than the password itself. Thus, we also call it the implicit transmission of passwords. The password amplification idea is very similar to zero-knowledge proof because of such a property.

The Diffie-Hellman key agreement scheme needs two large exponentials, g^x and g^y, to be exchanged between parties. For the amplification of the password entropy, we conceal the password information into the exchanged messages by the well-defined functions such as $\mathcal{G}()$ and $\mathcal{H}()$. The following definition gives our conceptual model. The function $\mathcal{G}()$ is for the commitment and challenge while the function $\mathcal{H}()$ is for the response.

Definition 2 *The conceptual model of* AMP *defines; Alice who knows υ, says hello to Bob and asks if he knows ν by sending $\mathcal{G}_1(g^x)$ but Bob asks Alice with $\mathcal{G}_2(g^y)$ whether she knows υ. Alice replies with $\mathcal{H}_1(g^{yx})$ that she knows it. Bob replies with $\mathcal{H}_2(g^{xy})$ that he knows ν. Finally, they agree on the Diffie-Hellman exponential g^{xy} and authenticate each other.*

$$\begin{array}{rcl}
\text{Alice} & & \text{Bob} \\
x \in_R Z_q^*\ ,\ \upsilon & & y \in_R Z_q^*\ ,\ \nu \\
\textit{hello, do you know } \nu? & \xrightarrow{\mathcal{G}_1(g^x)} & \\
 & \xleftarrow{\mathcal{G}_2(g^y)} & \textit{well, do you know } \upsilon? \\
\textit{yes, I know it.} & \xrightarrow{\mathcal{H}_1(g^{yx})} & \\
 & \xleftarrow{\mathcal{H}_2(g^{xy})} & \textit{I know } \nu.
\end{array}$$

We assume *Alice* knows υ by giving Assumption 1 though she actually knows only π.

Assumption 1 *Salt τ is disclosed in every protocol run.*

For example, τ is retained by *Bob* but transmitted to *Alice* in every protocol run. We can also allow implicit salt by making both parties get τ respectively, e.g., from their identities[5], but it is not favorable to upgrading the existing salt systems, e.g. Unix password file.

The Function. Each party commits a secret and challenges the other party by sending $\mathcal{G}_1()$ or $\mathcal{G}_2()$. They reply to each other with $\mathcal{H}_1()$ or $\mathcal{H}_2()$. $\mathcal{G}_1()$ and $\mathcal{G}_2()$ should convey the high entropy information whereas $\mathcal{H}_1()$ and $\mathcal{H}_2()$ must transmit the verification information without leaking the sensitive information such as an agreed key and a password. $\mathcal{G}_2()$ is much more difficult to design than $\mathcal{G}_1()$ because *Alice* must show her verification information $\mathcal{H}_1()$ ahead of *Bob*. That is, *Bob* is always given the opportunity to analyze $\mathcal{G}_1()$, $\mathcal{G}_2()$, and $\mathcal{H}_1()$ without showing his verification information, e.g., a chosen exponent attack[3]. $\mathcal{H}_1()$ and $\mathcal{H}_2()$ are to be designed by a strong one-way hash function. We discuss the details in section 3.

2.2 A Communication Model

This section formalizes the communication model of our protocol. Our communication model is based on the work of Bellare and Rogaway[3], and the following work of Stefan Lucks[22]. That is, all communication among interacting parties is under the adversary's control.

The Protocol. The protocol can be formally specified by an efficiently computable function Π for two players; let us set $I \in \{A, B\}$ for *Alice* and *Bob*. Note that *Eve* is not included in the players[3]. Each party can be formally modeled by an infinite collection of oracles.

Definition 3 *Our protocol is a set of function* $\Pi_I(1^k, \sigma, \kappa, r) = (m, \delta, \mu)$ *for* I.

- 1^k : the security parameter, $k \in \mathcal{N}$.
- $\sigma \in \{0,1\}^*$: the secret information of the sender.
- $\kappa \in \{0,1\}^*$: the conversation so far.
- $r \in \{0,1\}^\infty$: the random coin flips of the sender.
- $m \in \{0,1\}^* \cup \{*\}$: the next message for a reply.
- $\delta \in \{Accept, Reject, *\}$: the decision.
- $\mu \in \{0,1\}^* \cup \{*\}$: the agreed session key.

The value $\{*\}$ means; (1) there is no reply message for m, (2) the decision is not yet made for δ, (3) the decision is neither yet made nor accepted for μ.

[3] An adversary chooses an exponent and forges the message for further off-line verifications.

The Weak-Key Generator. A weak-key generator is $\mathcal{W}(1^{\omega(k)}, \iota, r_G)$ where $\iota \in I \cup \{E\}$ and $r_G \in \{0,1\}^\infty$. Note that the weak-key has a length k but its entropy is totally different[4]. When we assume the strong-key length is just k, i.e. our security parameter, the parameter $\omega(k)$ means the low entropy of the weak-key. Brute-forcing $2^{\omega(k)}$ values is feasible whereas k is large enough against brute-forcing 2^k values (hence $2^{\omega(k)} \ll 2^k$). The point of the weak-key is that the adversary is denied by the generator as like the long-lived key case[3] but repeatedly it is acceptable in the probability of $2^{-\omega(k)}$. Our model agrees on $\mathcal{W}(1^{\omega(k)}, A, r_G) = \upsilon$, $\mathcal{W}(1^{\omega(k)}, B, r_G) = \nu$, and $\mathcal{W}(1^{\omega(k)}, E, r_G) = \lambda$ under Assumption 1. The idea is typical in that the adversary has the probability of $2^{-\omega(k)}$ for guessing the weak key.

The Adversary. The adversary Eve is represented as a probabilistic machine $\mathcal{E}(1^k, \sigma_E, r_E)$ equipped with an infinite collection of oracles Π_I^s for $i \in I$ and $s \in \mathcal{N}$[3]. When the adversary is deterministic and restricts its action to faithfully conveying each flow from one oracle to the other, i.e., matching conversations, she is called a benign adversary[3]. $\mathcal{E}$ communicates with the oracles via queries of the form $\mathcal{Q}(i, s, n)$; Eve sends message n to the oracle of i. If the oracle Π_i^s has accepted, Eve is able to ask Π_i^s for its session key by sending some special query such as $\mathcal{Q}(i, s, reveal)$. The query will be answered by Π_i^s by the following experiment.

Running The Protocol. Running a protocol Π (with the weak-key generator $\mathcal{W}$) in the presence of Eve, using security parameter k, means performing the following experiment:

1. Choose a random string $r_G \in \{0,1\}^\infty$ and $\sigma_i = \mathcal{W}(1^{\omega(k)}, i, r_G)$, for $i \in I$, and set $\sigma_E = \mathcal{W}(1^{\omega(k)}, E, r_G)$.

2. Choose a random string $r_i^s \in \{0,1\}^\infty$ for $i \in I$ and $s \in \mathcal{N}$, and a random string $r_E \in \{0,1\}^\infty$.

3. Let $\kappa_i^s = \lambda$ for all $i \in I$ and $s \in \mathcal{N}$.

4. Run the adversary, $\mathcal{E}(1^k, \sigma_E, r_E)$, answering oracle calls as follows. When $\mathcal{E}$ asks a query, $\mathcal{Q}(i, s, n)$, oracle Π_i^s answers with (m, δ) by computing $(m, \delta, \mu) = \Pi_I(1^k, \sigma, \kappa_i^s.n, r_i^s)$, and sets $\kappa_i^s = \kappa_i^s.n$.

5. The adversary chooses an oracle Π_i^s and attempts to guess its session key or weak-key.

2.3 Security Definition

This subsection addresses the negligible probability and defines the secure protocol on this premise.

[4] Actually the weak-key, i.e. the password, is chosen by a user through a limited input device such as a keyboard or keypad, and in a small set of human-memorable word space.

Negligible Probability. We already mentioned and assumed the security parameter k. It is long enough to prevent the brute-forcing on itself. Therefore, we simply define the negligible probability, $\mathcal{P}[] \leq 2^{-k}$.

Secure Protocol. We create a rigorous condition for the protocol to provide the server-compromise resistant password authentication and key agreement. As already discussed, there are a passive(benign) adversary and an active adversary. Our model allows either of them to analyze the information given by the oracle.

Definition 4 *Protocol* Π *is a* **secure authenticated key exchange with a weak-key generator** $\mathcal{W}()$ *if the following statements are true:*

1. If two oracles have matching conversations, then both oracles accept and agree on the identical key.
2. If it is the case that the oracles accept and agree on the same key, then the probability of no-matching is negligible.
3. If *Eve* is benign, her probability of success is negligible.
4. If *Eve* has been rejected R times, the possible set of the guessing π decreases linearly, $2^{\omega(k)} - R$.
5. If *Eve* has been rejected $R(< 2^{\omega(k)} - 1)$ times but finally remains benign, her probability of success is still negligible.

Note that the success of *Eve* means she has been accepted by the oracle.

The first condition has to do with the completeness of the protocol. That is, if each party's messages are faithfully relayed to one another, then the parties succeed in authentication and key agreement, at least with overwhelming probability.

The second condition implies that any adversaries cannot be accepted by the oracle without knowing the password (the verifier for *Bob*) and the agreed key.

The third condition regards the security against the passive adversary who always listens to the conversation and analyzes the eavesdropped messages.

The fourth condition holds that an on-line trial cannot partition out the possible set. The partitioning implies that the possible set decreases logarithmically. Note that an on-line success is quite negligible if we count the failed-trials and cut off the exceeding trial.

The fifth condition deals with security against the active adversary who occasionally participates in the conversation but mostly analyzes the gathered information off-line. This facilitates systematic analysis of the reply to the chosen challenge.

Finally note that the first three conditions are about the accepted session whereas the last two conditions are for the rejected session.

3 Protocol Construction

This section summarizes the concrete design of AMP and examines the security of AMP on the random oracle model. We will give a practical analysis and a rigorous comparison to others in section 4. The protocol described in the following, is based on the model of Definition 2.

3.1 AMP Design

The followings are describing the setup and the run of our protocol.

Random Oracle. We assume random oracles $h_i() : \{0,1\}^* \rightarrow \{0,1\}^{l(k)}$ for $i \in [1,4]$. If *Eve* sends queries $\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3, ...$ to the random oracle h_i, she can receive answers $h_i(Q_j)$, all independently random values, from the oracle. Note that $\varphi()$ is in $\{0,1\}^{\leq 2l(k)}$ while $h()$ is in $\{0,1\}^{l(k)}$. Therefore, the inequality, $2l(k) > l(k)$, allows us to regard all of them $h_i()$ for simplicity. We use the random oracles $h_1()$ and $h_2()$ for secure exponent and secure session key, respectively, while we use $h_3()$ and $h_4()$ for $\mathcal{H}_1$ and $\mathcal{H}_2$.

Numerical Assumption. We assume that all numerical operations of the protocol are on the cyclic group where it is hard to solve the Diffie-Hellman problem as well as the discrete logarithm problem. We consider the multiplicative group Z_p^* and actually use its prime-order subgroup Z_q^*. Note that we should use only its main operation, a modular multiplication, for easy generalization. For the purpose, *Bob* chooses g which generates a prime-order subgroup Z_q^* where $p = qr + 1$. Note that a prime q must be sufficiently large ($> l(k)$) to resist against various index-calculus methods but much smaller than p[28, 29, 26]. It is easy to make g by $\alpha^{(p-1)/q}$ where α generates Z_p^*. Z_p^* is also applicable but Z_q^* is preferred for efficiency and for preventing a small subgroup confinement more effectively. By confining all exponentiation to the large prime-order subgroup through g of Z_q^*, each part of the protocol is able to detect on-line attack whenever a received exponenital is confined to a samll subgroup. The use of a safe prime, $p = 2q+1$, is the extreme case so that it is only recommended.

Protocol Setup. This step determines and publishes global parameters of AMP.

1. *Alice* and *Bob* shares g, p and q.
2. *Alice* chooses $\pi \in_R \{0,1\}^{\omega(k)}$ and notify *Bob*, in an authentic manner[5].
3. *Bob* chooses $\tau \in_R \{0,1\}^{t(k)}$ and stores $(id, \tau, \nu = g^{\upsilon})$ where $\upsilon = h_1(\tau, \pi)$.
4. *id* indicates an identifier or name of *Alice*; more precisely a user name.

[5] We imply by "an authentic manner" that the corresponding parameters are shared by some safe method, for example, secure registration, off-line distribution or picture id proof.

Bob should throw away π and υ but keep τ and ν. Once *Alice* and *Bob* agree upon π and ν, they can run the following.

PROTOCOL RUN. The following describes how to run AMP. Note that the cases, $x \in \{0,1\}^1$, $y \in \{0,1\}^1$, $g^x \in \{0,1\}^1$, $(g^x\nu)^y \in \{0,1\}^1$, and their small subgroup confinements must be avoided for a security reason. *Alice* and *Bob* can easily detect and discard such insecure parameters in the protocol.

Alice		Bob
$input\ (id, \pi)$		$store\ (id, \tau, \nu)$
$x \in_R Z_q^*$		
$\mathcal{G}_1 = g^x$	$\xrightarrow{id,g^x}$	
		$fetch\ (id, \tau, \nu)$
		$y \in_R Z_q^*$
	$\xleftarrow{\tau,(g^x\nu)^y}$	$\mathcal{G}_2 = (\mathcal{G}_1\nu)^y$
$\upsilon = h_1(\tau, \pi)$		
$\chi = (x+\upsilon)^{-1}x \bmod q$		
$\alpha = (\mathcal{G}_2)^{\chi}$		$\beta = (\mathcal{G}_1)^y$
$\mathcal{K}_1 = h_2(\alpha)$		$\mathcal{K}_2 = h_2(\beta)$
$\mathcal{H}_{11} = h_3(\mathcal{G}_1, \mathcal{K}_1)$	$\xrightarrow{h(g^x,\mathcal{K})}$	$\mathcal{H}_{12} = h_3(\mathcal{G}_1, \mathcal{K}_2)$
		$verify\ \mathcal{H}_{11} \doteq \mathcal{H}_{12}$
$\mathcal{H}_{21} = h_4(\mathcal{G}_2, \mathcal{K}_1)$	$\xleftarrow{h((g^x\nu)^y,\mathcal{K})}$	$\mathcal{H}_{22} = h_4(\mathcal{G}_2, \mathcal{K}_2)$
$verify\ \mathcal{H}_{21} \doteq \mathcal{H}_{22}$		

Figure 1. AMP Protocol

The following steps explain how the protocol is executed in Figure 1.

1. *Alice* computes $\mathcal{G}_1 = g^x$ by choosing $x \in_R Z_q^*$ and sends it with her id to *Bob*.

2. After receiving message 1, *Bob* loads τ and ν, and computes $\mathcal{G}_2 = (\mathcal{G}_1\nu)^y$ by choosing $y \in_R Z_q^*$. This can be done by the simultaneous exponentiation method, i.e., ${\mathcal{G}_1}^y\nu^y$. Note that $\mathcal{G}_2 = (g^x\nu)^y = g^{(x+\upsilon)y}$. He sends it with τ to *Alice*.

3. After receiving message 2, *Alice* computes $\upsilon = h_1(\tau, \pi)$ and $\alpha = (\mathcal{G}_2)^{(x+\upsilon)^{-1}x}$ by obtaining $(x+\upsilon)^{-1} \bmod q$. Note that $\alpha = (g^{(x+\upsilon)y})^{(x+\upsilon)^{-1}x} = (g^y)^x = g^{yx}$. She computes $\mathcal{K}_1 = h_2(\alpha)$ and $h(\mathcal{G}_1, \mathcal{K}_1)$. She sends $h(\mathcal{G}_1, \mathcal{K}_1)$ to *Bob*.

4. While waiting for message 3 from *Alice*, *Bob* computes $\beta = (g^x)^y = g^{xy}$, $\mathcal{K}_2 = h_2(\beta)$ and $h_3(\mathcal{G}_1, \mathcal{K}_2)$. After receiving message 3, *Bob* compares $h_3(\mathcal{G}_1, \mathcal{K}_1)$ with $h_3(\mathcal{G}_1, \mathcal{K}_2)$. If $h_3(\mathcal{G}_1, \mathcal{K}_1) = h_3(\mathcal{G}_1, \mathcal{K}_2)$, then he computes $h_4(\mathcal{G}_2, \mathcal{K}_2)$ and sends it to *Alice*. This means he authenticated *Alice* who knows υ (actually, π), and agreed upon $\mathcal{K}(= \mathcal{K}_1 = \mathcal{K}_2)$.

5. While waiting for message 4 from *Bob*, *Alice* computes $h_4(\mathcal{G}_2, \mathcal{K}_1)$. After receiving message 4, she compares $h_4(\mathcal{G}_2, \mathcal{K}_1)$ with $h_4(\mathcal{G}_2, \mathcal{K}_2)$. If $h_4(\mathcal{G}_2, \mathcal{K}_1) = h_4(\mathcal{G}_2, \mathcal{K}_2)$, *Alice* also agrees on $\mathcal{K}(= \mathcal{K}_1 = \mathcal{K}_2)$ with authenticating *Bob* who knows ν.

AMP passes four messages between *Alice* and *Bob*. Each party computes the exponentiation, $O((\log n)^3)$, only for two times, respectively. Note that $\mathcal{G}_1() = \mathcal{G}_1$ and $\mathcal{G}_2() = \mathcal{G}_2$ while $\mathcal{H}_1() = \mathcal{H}_{1i}$ and $\mathcal{H}_2() = \mathcal{H}_{2i}$ for $i \in \{1, 2\}$. i can be abbreviated for simplicity.

3.2 Security Examination

The following lemma shows that it is infeasible for *Eve* to impersonate *Alice* or *Bob* by forging $\mathcal{G}_1$ or $\mathcal{G}_2$ even with off-line verifications. Note that, when *Eve* forges $\mathcal{G}_2$, she is given the information such as $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{H}_1$. That is, she is allowed to attempt to verify υ off line. The success of off-line verification implies the acceptance of $\mathcal{H}_1$ or $\mathcal{H}_2$ in the next session by getting υ, while that of on-line verification means the acceptance in the current session.

Lemma 1 *The probability of success by forging $\mathcal{G}_1$ or $\mathcal{G}_2$ is negligible in* AMP.

Proof Sketch: We consider each of $\mathcal{G}_1$ and $\mathcal{G}_2$. There could be three cases on the random oracle model. For the success, each $\mathcal{H}$ must be correctly generated and verified by the oracle.

1. Let *Eve* forge $\mathcal{G}_1$ by choosing $x \in_R Z_q^*$ and making $\mathcal{G}_1^{'} = g^x$. Then *Bob* replies with $\mathcal{G}_2 = g^{(x+\upsilon)y}$. *Eve* could find $\alpha^{'} = g^{y'x}$ by computing ${\mathcal{G}_2}^{(x+\upsilon')^{-1}x}$ with $\upsilon^{'} \in_R \{0,1\}^{\omega(k)}$. However, she cannot verify $\alpha^{'} \doteq \beta$, i.e., $g^{y'x} \doteq g^{xy}$, without submitting $\mathcal{H}_1^{'}$ ahead of $\mathcal{H}_2$. The probability of successful submission is $2^{-\omega(k)}$. It can be easily detected by *Bob* in only R trials where R is a pre-defined very small integer such that $R << 2^{\omega(k)/2}$.

2. Let *Eve* forge $\mathcal{G}_2$ by choosing $c \in_R Z_q^*$ and making $\mathcal{G}_2^{'} = g^c$. Then *Alice* replies with $\mathcal{H}_1$ by getting $\alpha = (\mathcal{G}_2^{'})^{(x+\upsilon)^{-1}x}$. We can rewrite $c = (x + \upsilon^{'})y^{'} \bmod q$ where $\upsilon^{'}$ and $y^{'}$ are variables. *Eve* must find $y^{'}$ such that $y^{'} \equiv c(x + \upsilon^{'})^{-1} (\text{mod } q)$ for getting $\beta^{'} = (\mathcal{G}_1)^{y'}$ and verifying $\alpha \doteq \beta^{'}$, i.e., $\mathcal{H}_{11} \doteq \mathcal{H}_{12}^{'}$. $\beta^{'}$ is also necessary for making $\mathcal{H}_2^{'}$. For the computation of $y^{'}$, it is necessary to know x of $\mathcal{G}_1$. However, x is an unknown variable chosen in $\{0,1\}^{\geq l(k)}$ randomly, while finding x from $\mathcal{G}_1$ is the discrete logarithm problem. Therefore, the probability of finding and verifying $\alpha \doteq \beta^{'}$ is $\mathcal{P}[] < 2^{-k}$.

3. Let *Eve* forge $\mathcal{G}_2$ by choosing $y \in_R Z_q^*$ and $\upsilon^{'} \in_R \{0,1\}^{\omega(k)}$, and making $\mathcal{G}_2^{'} = (\mathcal{G}_1 g^{\upsilon'})^y$. Then *Alice* replies with $\mathcal{H}_1$ by computing $\alpha = (\mathcal{G}_2^{'})^{(x+\upsilon)^{-1}x}$, but note that $\alpha = (g^{(x+\upsilon')}y)^{(x+\upsilon)^{-1}x}$. That is, $\alpha = g^{y'x}$ rather than g^{yx} where $y^{'} \equiv (x + \upsilon')y(x + \upsilon)^{-1} (\text{mod } q)$. Finding $y^{'}$ or υ is the only way for *Eve* to be accepted by the oracle.

(a) x chosen by *Alice*, $x \in_R \{0,1\}^{\geq l(k)}$, is not known to *Eve*. We can say that $v = v^{'}$ if and only if $y^{'} \equiv y (\bmod\ q)$. It corresponds to the on-line attack.

   i. Let *Eve* attempt to verify $v \doteq v^{'}$. However, $v^{'}$ is rather a constant because she defined it before receiving $\mathcal{H}_1$ from *Alice*. *Eve* cannot replace $v^{'}$ for further verification without retrying it on line. The probability of $v = v^{'}$ is $2^{-\omega(k)}$; an extremely low probability for on-line success. Due to the maximum count of on-line failure, R, she must be denied by the oracle before trying $2^{\omega(k)} - R$ more guesses. The probability of $v \neq v^{'}$ is very high such that $\mathcal{P}[] \leq 1 - \frac{1}{2^{\omega(k)} - R}$. Therefore, we can say hereafter $v^{'}$ is a constant such that $v \neq v^{'}$ by following the weak-key generator $\mathcal{W}()$ such that $\mathcal{W}(1^{\omega(k)}, E, r_G) = \lambda$.

   ii. Let *Eve* attempt to find $y^{'}$. However, she has to know x for attempting the equation, $y^{'} \equiv (x + v')y(x + v)^{-1} (\bmod\ q)$. Finding x from $\mathcal{G}_1$ is the discrete logarithm problem. Even if *Eve* computes $\alpha^{'} = (\mathcal{G}_1)^y = (g^x)^y$, it is clear that $\alpha \neq \alpha^{'}$ when $v \neq v^{'}$. Therefore, the probability of finding $y^{'}$ is $\mathcal{P}[] < 2^{-k}$.

(b) Let *Eve* guess $v^{''} \in_R \{0,1\}^{\omega(k)}$ and compute $\alpha^{''} = \mathcal{G}_2^{'} g^{-v''y} = g^{(x+v')y - v''y}$ for verifying $\alpha^{''} \doteq \alpha$ in $2^{-\omega(k)}$ probability, rather than attempt to replace $v^{'}$ with $v^{''}$. If $\alpha^{''} = \alpha$ with guessed $v^{''}$, she can be convinced $v = v^{''}$. Thus, if the equation, $(x + v^{'})y(x+v)^{-1}x \equiv xy + v^{'}y - v^{''}y (\bmod\ q)$ is true, then she can find v in $2^{-\omega(k)}$ probability, regardless of x and $v^{'}$. Otherwise, she has to find x first. We can rewrite it as $(x + v^{'})\ \cancel{y}(x + v)^{-1}\ \cancel{x} \equiv \cancel{x}\ \cancel{y}(1 + v^{'}x^{-1} - v^{''}x^{-1})(\bmod\ q)$. That is, $(x+v^{'})(x+v)^{-1} \equiv (1+v^{'}x^{-1} - v^{''}x^{-1})(\bmod\ q)$. We can transpose $(x+v)^{-1}$ so that $(x+v^{'}) \equiv (x+v) + (x+v)v^{'}x^{-1} - (x+v)v^{''}x^{-1} \equiv x+v+v'+vv^{'}x^{-1} - v^{''} - vv^{''}x^{-1} \equiv (x + v^{'}) + (v - v^{''}) + (v^{'} - v^{''})vx^{-1}(\bmod\ q)$. Then, we can transpose $(x + v^{'})$ so that $0 \equiv (v - v^{''}) + (v^{'} - v^{''})vx^{-1}(\bmod\ q)$. Therefore, the equality such that $v = v^{'} = v^{''}$ (due to $v = v^{''}$ and $v^{'} = v^{''}$), is the mandatory requiremet of this modular equation. However, the probability of success is negligible because $v \neq v^{'}$ with very high probability as we mentioned above in (a). Therefore, *Eve* must find x for getting v. The probability of verifying $\alpha^{''} \doteq \alpha$ is $\mathcal{P}[] < 2^{-k}$.

After all, the probability of success is negligible, because on-line retrial is detectable while off-line verification has a probability such that $\mathcal{P}[] < 2^{-k}$. □

The following theorem shows AMP is a secure authenticated key exchange protocol with a weak-key generator $\mathcal{W}()$. Security against conventional attacks is examined in section 4.

Theorem 1 AMP *is a secure authenticated key exchange with a weak-key generator* $\mathcal{W}()$.

Proof Sketch: To be accepted by the oracle, *Eve* must get v, ν or $\mathcal{K}$. We show a probability is negligible by following each condition of Definition 4 on the random oracle model.

1. A completeness of the protocol is already proved in Figure 1 and its description. If two oracles have matching conversations, $\mathcal{H}_1$ and $\mathcal{H}_2$ can be verified since $\alpha = \beta$ $(\mathcal{K}_1 = \mathcal{K}_2)$.

2. The final acceptance by the oracles means both $\mathcal{H}_1$ and $\mathcal{H}_2$ are successfully verified. That is, $h_3(\mathcal{G}_1, \mathcal{K}_1) = h_3(\mathcal{G}_1, \mathcal{K}_2)$ and $h_4(\mathcal{G}_2, \mathcal{K}_1) = h_4(\mathcal{G}_2, \mathcal{K}_2)$. We show the probability of no-matching is always $\mathcal{P}[] \leq 2^{-k}$ when it is accepted by the oracles.

   (a) The birthday paradox is negligible due to the nature of the random oracle, $h_i() : \{0,1\}^* \rightarrow \{0,1\}^{l(k)}$; the probability is $2^{-\frac{1}{2}l(k)} \leq 2^{-k}$.

   (b) Due to item (a), the correct value $\mathcal{K}(= \mathcal{K}_1 = \mathcal{K}_2)$ is mandatory for the acceptance even without matching conversations. For finding $\mathcal{K}$, *Eve* must obtain α or β.

      i. The probability of guessing $g^{xy}(= \alpha = \beta)$ with nothing is surely $\mathcal{P}[] < 2^{-k}$.

      ii. We can rewrite α and β where $\mathcal{G}_1 = g^{\log \mathcal{G}_1}$ and $\mathcal{G}_2 = (\mathcal{G}_1 \nu)^{(\log \mathcal{G}_1 + v)^{-1} \log \mathcal{G}_2}$. That is, $\alpha = (\mathcal{G}_2)^{(\log \mathcal{G}_1 + v)^{-1} \log \mathcal{G}_1} = (\mathcal{G}_1)^{(\log \mathcal{G}_1 + v)^{-1} \log \mathcal{G}_2} = \beta$. log means $\log_g$. Thus, we can find easily the flows, $(\mathcal{G}_1 \rightarrow \mathcal{G}_2 \rightarrow \alpha)$ and $(\mathcal{G}_1 \rightarrow \mathcal{G}_2 \rightarrow \beta)$. Without such flows, the exponents of $\mathcal{G}_1$ and $\mathcal{G}_2$ must be analyzed but the probability is $\mathcal{P}[] \leq 2^{-l(k)}$ even with a forged attempt by Lemma 1.

3. When *Eve* is benign, all she receives from the oracle are $\{id, \tau, \mathcal{G}_1, \mathcal{G}_2, \mathcal{H}_1, \mathcal{H}_2\}$ for every session where their internal values, x and y, are independent random values from $\{0,1\}^{\leq 2l(k)}$. Therefore, g^x, g^y, and their composition on the cyclic group are well distributed on the group. We assume the uniform distribution. Since $\mathcal{W}(E) = \lambda$ and $\mathcal{G}_2 = (\mathcal{G}_1 g^v)^y$, the probability of finding $g^{y'}$ by guessing π' in $\mathcal{G}_1$ and $\mathcal{G}_2$, is less than $2^{-(\omega(k)+l(k))}$. For verifying the guess of π', *Eve* must find $\mathcal{K}$ for asking the random oracle. However, finding $\mathcal{K}$ over g^x and g^y is the Diffie-Hellman problem. Therefore, the probability of success for benign *Eve* is $\mathcal{P}[] < 2^{-k}$.

4. Since $\mathcal{G}_1$ and $\mathcal{G}_2$ remain on the cyclic group under uniform distribution and the random oracle h_1 converts the password to the sufficient-size exponent, there is no way to find the relationship between the rejected password and the remaining password. Other possibilities are all negligible by Lemma 1. If *Eve* is rejected, she must reduce the set by one, $2^{\omega(k)} - 1$, and try again with another guess. That is, the set is reduced linearly. Therefore, the success probability of her on-line guess is only $\mathcal{P}[] \leq \frac{1}{2^{\omega(k)} - R - c}$ for very small $c (\geq 0)$. She must be denied by the oracle only in R trials as in Lemma 1.

5. Assume there is no detection count R and *Eve* has been rejected with different password guesses $2^{\omega(k)} - 2$ times by the oracle, then she could have bernoulli trial on two remaining passwords; if one is rejected then the other is the one and vice versa. However, if she remains benign even with bernoulli trial on guess, then she cannot ask the oracle any more. Therefore, she is only able to analyze the old rejected messages or the new eavesdropped messages. For the former case, the probability is less than 2^{-k} by Lemma 1. For the latter case, the probability is $2^{-(l(k)+1)} (< 2^{-k})$ by item 3 of this proof. Hence, the probability of success for partially benign *Eve* is negligible $(< 2^{-k})$.

AMP is a secure authenticated key exchange protocol with a weak-key generator $\mathcal{W}()$. □

3.3 Small Discussion

As we mentioned in section 2.1, we can remove τ from message 2 for implicit salt. We assumed all hash functions to behave like random oracles for security proof[3], and we supposed $\varphi()$ in $\{0,1\}^{\leq 2l(k)}$ as $h_1()$ in $\{0,1\}^{l(k)}$. However, they must be recovered in a real world. Let h() denote a hash function. Then, by following the constructions of the work[3] and recommending SHA-1 or RIPEMD-160 for 160-bit hash, we define; $h_1(x) = h(00|x)$, $h_2(x) = h(01|x)$, $h_3(x) = h(10|x)$, and $h_4(x) = h(11|x)$. $|$ denotes the concatenation. If we use 128-bit hash functions, we define $\varphi(x) = h(00|x|00|x)h(00|x|01|x)$ only instead of $h_1(x)$.

4 Analysis and Comparison

This section examines the security against conventional attacks and discusses the efficiency and constraints of AMP, while comparing to other functionally-similar (secure) protocols such as A-EKE, B-SPEKE, SRP, GXY, SNAPI-X, AuthA and PAK-X[7, 18, 34, 20, 23, 5, 9].

4.1 Security of AMP

Lemma 1 and Theorem 1 gave us the provable features of security. Let us ensure it.

1. AMP provides perfect forward secrecy via the Diffie-Hellman problem. That is, even if π (or v) is compromised, *Eve* cannot find old session keys because she is not able to solve the Diffie-Hellman problem. Condition 3 supports this feature.

2. Denning-Sacco attack is the case that *Eve*, who compromised an old session key, attempts to find π or to make the oracle accept her[10]. For the purpose, *Eve* also has to solve the Diffie-Hellman problem. Moreover, the actual session key is computed by $\mathcal{K} = h_2(g^{xy})$. Therefore, a compromised old session key is not helpful for *Eve* in attempting the Denning-Sacco attack. This feature is also supported by Condition 3.

3. Replay attack is negligible because $\mathcal{G}_1$ should include an ephemeral parameter of *Alice* while the others such as $\mathcal{G}_2$, $\mathcal{H}_1$ and $\mathcal{H}_2$, should include ephemeral parameters of both parties of the session. Finding those parameters is similar to solving the discrete logarithm problem and each parameter is bounded by $2^{-l(k)} < 2^{-k}$. Therefore, both active replay and succeeding verification are negligible. Condition 2 and 5 support this feature.

4. Small subgroup confinement is defeated and avoided by confining to the large prime-order subgroup. An intentional small subgroup confinement can be detected easily.

5. On-line guessing attack is detectable and the following off-line analysis can be frustrated, even if *Eve* attempts to disguise *Alice* or *Bob*. Impersonation of the party or man-in-the-middle attack is also infeasible without knowing v or ν. Condition 2 and 4 support the on-line detection while Lemma 1 and Condition 5 support the latter off-line frustration.

6. Off-line guessing attack is also infeasible by Condition 3 and 5. Partition attack is to reduce the set of passwords logarithmically by asking the oracle in parallel with off-line analysis, while chosen exponent attack is to analyze it via her chosen exponent. Those attacks are frustrated by Condition 4 and Lemma 1, respectively.

7. Security against password-file compromise is the basic property of AMP. Definition 1 and Condition 1 support this feature.

In conclusion, AMP is secure as we expected.

4.2 Efficiency and Constraints

Performance of these protocol families can be approximated in terms of communication and computation loads (see Table 1). We summarize the efficiency and constraints of AMP.

Efficiency. The efficiency of AMP can be discussed as follows.

1. In the aspect of a communication load, AMP has only four protocol steps while the number of large message blocks is only two in AMP. They are $\mathcal{G}_1$ and $\mathcal{G}_2$.

2. A total amount of execution time could be approximated by the number of modular exponentiation by considering the parallel execution of both parties. We describe it as $E(Alice : Bob)$. Note that AMP has only 3E totally. They are $E(g^x : -)$, $E(- : (\mathcal{G}_1\nu)^y)$ and $E((\mathcal{G}_2)^\chi : (\mathcal{G}_1)^y)$. Here '$-$' means no exponentiations.

3. Each party of AMP performs only two exponentiations, respectively. It is the same to the Diffie-Hellman scheme though *Bob* does not use a small base g directly, i.e., $(\mathcal{G}_1\nu)^y$.

4. For run time parameters, each party generates only one random number, respectively. *Alice* can reduce her run time exponentiations to only one and parallel exponentiations to only two, by pre-computation of g^x.

5. $\mathcal{G}_2$ can benifit from the simultaneous exponentiation method as $(\mathcal{G}_1)^y\nu^y$. $\alpha^x\beta^y$ nees only 1.16 times more multi-precision multiplication than α^x does on the average[31, 24].

6. In step 3, *Alice* should compute $(y+v)^{-1}$ but only in the q-order subgroup. Modular inversion, $O((\log q)^2)$, is quite negligible against modular exponentiation, $O((\log p)^3)$.

7. AMP uses the main group operation so that it is *easy-to-generalize* in any cyclic groups. Therefore, AMP can be easily implemented on the elliptic curve group. A generalization on such a group must be very useful for further efficiency of space and speed, though there may a patent restriction.

A rigorous comparison to the other protocols is made in section 4.3

Constraints. AMP gives very light constraints as follows.

	Protocol	*Large*	*Exponentiations*			*Random Numbers*	
	Steps	*Blocks*	*Client*	*Server*	*Parallel*	*Client*	*Server*
A-EKE	7 (+4)	3 (+1)	4 (+2)	4 (+2)	6 (+3)	**1** (+0)	**1** (+0)
B-SPEKE	4 (+1)	3 (+1)	3 (+1)	4 (+2)	6 (+3)	**1** (+0)	2 (+1)
SRP	4 (+1)	**2** (+0)	3 (+1)	3 (+1)	4 (+1)	**1** (+0)	**1** (+0)
GXY	4 (+1)	**2** (+0)	4 (+2)	3 (+1)	5 (+2)	**1** (+0)	**1** (+0)
SNAPI-X	5 (+2)	5 (+3)	5 (+3)	4 (+2)	7 (+4)	2 (+1)	3 (+2)
AuthA	**3** (+0)	**2** (+0)	4 (+2)	3 (+1)	6 (+3)	**1** (+0)	**1** (+0)
PAK-X	**3** (+0)	3 (+1)	4 (+2)	4 (+2)	8 (+5)	**1** (+0)	2 (+1)
AMP	4 (+1)	**2** (+0)	**2** (+0)	**2** (+0)	**3** (+0)	**1** (+0)	**1** (+0)

Table 1: Comparisons of Verifier-based Protocols

1. AMP prefers g to be a generator of the large prime-order subgroup Z_q^* for defeating and avoiding a small subgroup confinement effectively by confining exponentials into the large prime-order subgroup[26]. A safe prime modulus is recommended as an extreme case for easy detection of an intentional small subgroup confinement.

2. A compromise of ν allows a guessing attack or a server impersonation but it is an inevitable feature of all verifier-based protocols[34].

3. AMP needs both parties to count the other side's on-line failure to detect the on-line guessing attack. However, this is the shared requirement of all password protocols.

4.3 Comparisons to Others

AMP is rigorously compared to the existing verifier-based protocols such as A-EKE, B-SPEKE, SRP, GXY, SNAPI-X, AuthA and PAK-X[7, 18, 34, 20, 23, 5, 9]. We disregard the security issue because all of them are believed secure.

Table 1 compares them with regard to several efficiency factors such as the number of protocol steps, large message blocks, and exponentiations. The random number is given as a subsidiary reference. Protocol steps and large blocks are critical factors to the communication load, while exponentiations and random numbers are to the computation load. The number of parallel exponentiations could compare approximately the amount of protocol execution time. The large block means a large cryptographic block based on the public-key cryptography such as the Diffie-Hellman exponential or the RSA message. The value in parenthesis implies the difference from the most efficient one that is denoted by bold characters.

As we can see in Table 1, AMP can be expected as the most efficient verifier-based protocol because it has the minimum values mostly. Actually, AMP is the most efficient compared to the others. Let us review the other protocols and discuss the superiority of AMP.

A-EKE. Bellovin and Merrit introduced the first verifier-based protocol, A-EKE in 1993[7]. In A-EKE, the symmetric encryption required p very close to 2^n where n is a bit-length of p, and random data to be padded to fill the block size[6]. A certified generator[27] and a digital signature scheme such as ElGamal [13] or p-NEW signature[25] were hard requirements. A-EKE has many protocol steps and exponentiations, and requires a safe prime modulus.

B-SPEKE. Jablon proposed B-SPEKE, the verifier-based augmentation of SPEKE, in 1997[17, 18]. Among the sets of B-SPEKE, the combined B-SPEKE is the most optimized so that it is compared in Table 1. However, it requires relatively quite heavy (parallel) exponentiations, and needs a server to choose another random number and a safe prime modulus. The parallel B-SPEKE allows four parallel exponentiations but it requires six message exchanges[18].

Srp. Wu proposed SRP that is notable in its practical approach, in 1998[34]. The benchmark shows its superiority to the above old schemes[34]. SRP does not agree to the Diffie-Hellman exponential and is not favorable to the generalization, especially in the elliptic curve group. It is not notable in its security proof. However, we can say SRP is a reasonably efficient and reliable verifier-based protocol (see Figure 2).

Gxy. Kwon and Song proposed GXY that is derived from SRP but agrees to the Diffie-Hellman exponential, in 1999[20]. However, it is less efficient than SRP. GXY is also not favorable to the generalization.

Snapi-X. MacKenzie and Swaminathan introduced the first provable verifier-based protocol, SNAPI-X, in 1999[23]. It was the augmented version of their basic protocol SNAPI. It is notable in its full proof of security by combining RSA and Diffie-Hellman scheme. However, it has the worst results for most factors in Table 1, and has additional constraints on choosing specific parameters and export/patent restrictions.

AuthA. Bellare, Pointcheval, and Rogaway introduced a provable approach, AuthA, by assuming an ideal cipher in 2000[5]. AuthA reduces her protocol steps without exchanging salt but must add two more steps to accomodate the existing salt schemes. Coincidentally, its message flow is very similar to KS-II[19], while its exponentiation sequences and contents are almost the same to GXY[20]. Table 1 also shows such a result.

Pak-X. Boyko, MacKenzie, and Patel introduced the provable verifier-based protocol, PAK-X, in 2000[9]. PAK-X is notable in its clearly provable approach. PAK-X is the last protocol in their three kinds of protocols but the only one that uses a verifier. It also reduces the number of protocol steps without exchanging salt. But instead, it is ranked as the worst

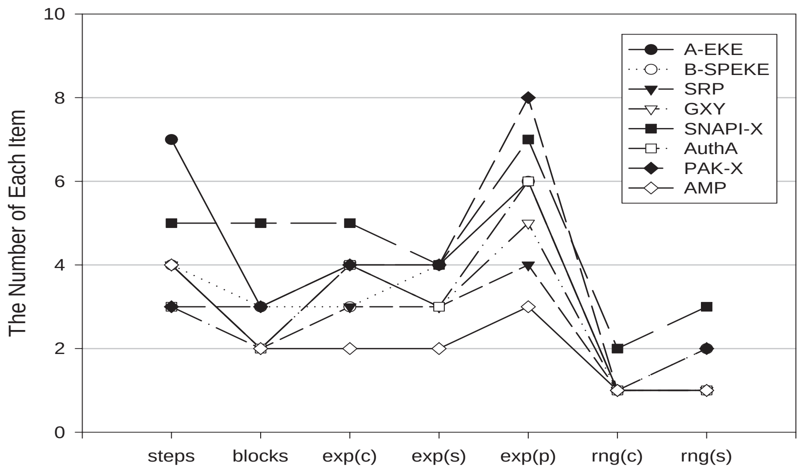

Figure 2. Graphical Representation of Table 1

performance protocol regarding the parallel exponentiations in Table 1.

WHY AMP. Figure 2 rewrites Table 1 graphically so that we can see AMP($\diamond$) has the best performance. The following items summarize well why AMP is believed the best in this paper.

1. AMP is a secure verifier-based protocol and it is provable on the random oracle model.
2. AMP is the most efficient protocol among the existing verifier-based protocols.
3. AMP has the least constraints and is easy to generalize, e.g., in elliptic curve groups.
4. AMP truly allows the Diffie-Hellman key agreement; (1) *Alice* sends g^x to *Bob* who simply raises it to y, i.e. g^{xy}; (2) *Bob* sends g^y to *Alice* by hiding it as $(g^y)^{(x+v)}$ while *Alice* obtains it by $((g^y)^{(x+v)})^{(x+v)^{-1}}$ and raises it to x, i.e. g^{yx}.
5. AMP is very simple in its structure, and accommodates any kinds of salt schemes so that the existing password file of various systems can be upgraded easily. Note that AuthA and PAK-X should add two more steps for this purpose.

5 Conclusion

In this paper, we introduced a new verifier-based protocol, AMP, for secure password authentication and the Diffie-Hellman key agreement, by following the previous notable methods. Compared to the similarly secure protocols, AMP was the most efficient one. AMP holds the provable security, the best efficiency, the light constraints, and the generalization features as its advantages. In addition, it allows an easy integration to the existing systems.

Internet business and commercial services are growing rapidly while personal privacy and security concerns are slower than those activities. Authentication is undoubtedly very important. Though the hardware-dependent authentication methods are growing steadily, the pure password authentication scheme is still the most reasonable in a distributed environment, and the public-key based cryptographic protocol is the best solution for improving its security. We should note that the only password authentication method can truly authenticate the human mind over the network. Therefore, we might keep utilizing it over the Internet and in mobile environments even with the hardware-supported authentication schemes.

References

[1] R.Anderson and T.Lomas, "Fortifying key negotiation schemes with poorly chosen passwords," Electronics Letters, vol.30, no.13, pp.1040-1041, 1994

[2] R.Anderson and S.Vaudenay, "Minding your p's and q's," Asiacrypt'96, LNCS, 1996

[3] M.Bellare and P.Rogaway, "Entity authentication and key distribution," Crypto 93, LNCS 773, 1993

[4] M.Bellare, R.Canetti, and H.Krawczyk, "A modular approach to the design and analysis of authentication and key exchange protocols," STOC 98, pp.419-428, 1998

[5] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attack," To appear in Eurocrypt 2000

[6] S.Bellovin and M.Merritt, "Encrypted key exchange : password-based protocols secure against dictionary attacks," IEEE Symposium on Research in Security and Privacy, pp. 72-84, 1992

[7] S.Bellovin and M.Merritt, "Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password-file compromise," ACM Conference on Computer and Communications Security, pp. 244-250, 1993

[8] M.Boyarsky, "Public-key cryptography and password protocols: the multi-user case," ACM Conference on Computer and Communication Security, September 16, 1999

[9] V. Boyko, P. MacKenzie and S. Patel, "Provably secure password authenticated key exchange using Diffie-Hellman," To appear in Eurocrypt 2000

[10] D.Denning, G.Sacco, "Timestamps in key distribution protocols," Communincations of ACM, vol.24, no.8, pp.533-536, 1981

[11] W.Diffie and M.Hellman, "New directions in cryptography," IEEE Transactions on Information Theory, vol.22, no.6, pp.644-654, Nov. 1976

[12] Y.Ding and P.Hoster, “Undetectable on-line password guessing attacks,” ACM Operating Systems Review, vol.29, no.4, pp.77-86, Oct. 1995

[13] T.Elgamal, “A public-key cryptosystem and a signature scheme based on discrete logarithms,” IEEE Trans. Information Theory, vol.IT-31, no.4, pp.469-472, 1985

[14] L.Gong, M.Lomas, R.Needham, and J.Saltzer, “Protecting poorly chosen secrets from guessing attacks,” IEEE Journal on SAC., vol.11, no.5, pp.648-656, June 1993

[15] L.Gong, “Optimal authentication protocols resistant to password guessing attacks,” IEEE Computer Security Foundation Workshop,, pp. 24-29 June 1995

[16] S.Halevi and H.Krawczyk, “Public-key cryptography and password protocols,” ACM Conference on Computer and Communications Security, 1998

[17] D.Jablon, “Strong password-only authenticated key exchange”, ACM Computer Communications Review, vol.26, no.5, pp.5-26, 1996

[18] D.Jablon, “Extended password key exchange protocols,” WETICE Workshop on Enterprise Security, 1997

[19] T.Kwon and J.Song, “Efficient key exchange and authentication protocols protecting weak secrets,” IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E81-A, no.1, pp.156-163, January 1998

[20] T.Kwon and J.Song, “Secure agreement scheme for g^{xy} via password authentication,” Electronics Letters, vol.35, no.11, pp.892-893, 27th May 1999

[21] M.Lomas, L.Gong, J.Saltzer, and R.Needham, “Reducing risks from poorly chosen keys,” ACM Symposium on Operating System Principles, ACM Operating Systems Review, pp.14-18, 1989

[22] S.Lucks, “Open key exchange: how to defeat dictionary attacks without encrypting public-keys,” The Security Protocol Workshop ’97, April 7-9, 1997

[23] P.MacKenzie amd R.Swaminathan, “Secure network authentication with password identification,” Presented to IEEE P1363a, August 1999

[24] A.Menezes, P.van Oorschot, S.Vanstone, *Handbook of applied cryptography*, CRC Press,Inc., 1997

[25] K.Nyberg and R.A.Rueppel, “Message recovery for signature scheme based on the discrete logarithm problem,” Eurocrypt 94, pp. 182-193, 1994

[26] P.van Oorschot and M.Wiener, “On Diffie-Hellman key agreement with short exponents,” Eurocrypt 96, pp. 332-343, 1996

[27] S.Patel, "Number theoretic attacks on secure password schemes," IEEE Symposium on Security and Privacy, 1997

[28] S.Pohlig and M.Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance," IEEE Transactions on Information Theory, vol.24, no.1, pp.106-110, 1978

[29] J.Pollard, "Monte carlo methods for index computation mod p," Mathematics of Computation, vol.32, pp.918-924, 1978

[30] M.Roe, B.Christianson, D.Wheeler, "Secure sessions from weak secrets," Technical report from University of Cambridge and University of Hertfordshire, 1998

[31] C.P.Schnorr, "Efficient identification and signatures for smart cards," Crypto 89, LNCS, pp.239-251, 1989

[32] M.Steiner, G.Tsudik, and M.Waidner, "Refinement and extension of encrypted key exchange," ACM Operating Systems Review, vol.29, no.3, pp.22-30, 1995

[33] G.Tsudik, E.van Herreweghen, "Some remarks on protecting weak keys and poorly-chosen secrets from guessing attacks," IEEE Computer Security Foundation Workshop, pp.136-142, 1993

[34] T.Wu, "Secure remote password protocol," Internet Society Symposium on Network and Distributed System Security, 1998

Appendix : Genealogy of Password Protocol

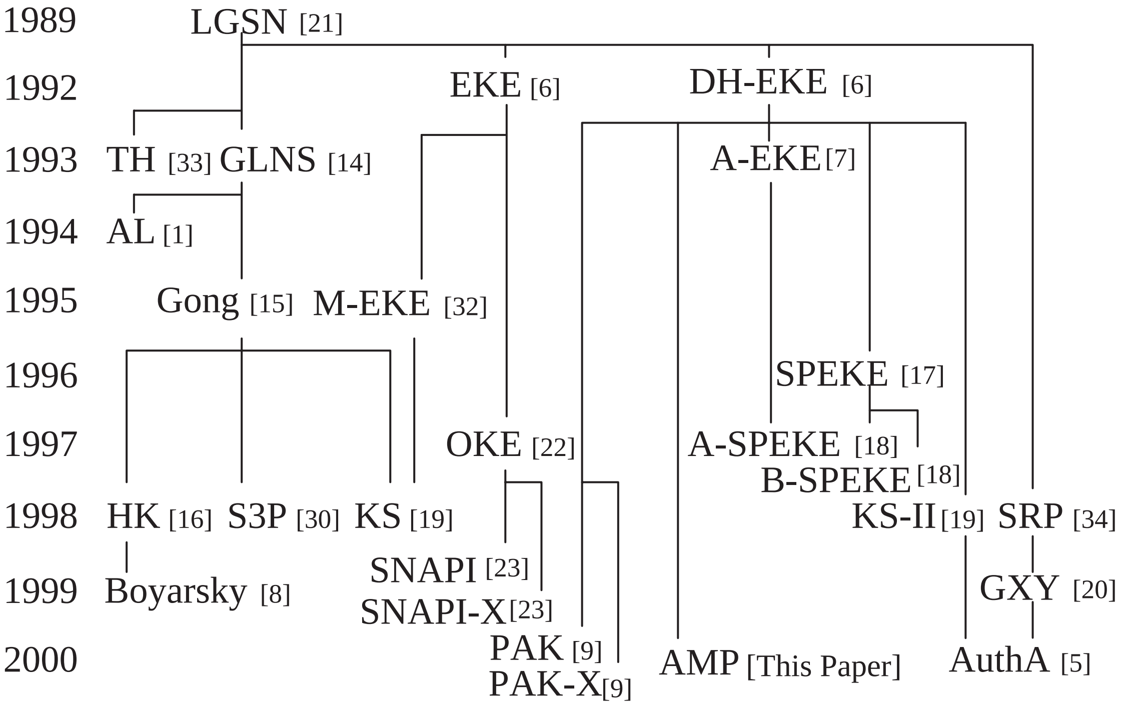

Figure 3. Password Authentication Protocols [6]

[6] The above genealogy is typically based on the opinion of the author. We analyzed all the protocols carefully and arranged them in the figure by considering their similarity or improvement. However, each author of the protocols could have different opinions. At this moment, we would like to make it clear that the above genealogy is only one of good references.