Authentication and Key Agreement via Memorable Password

Taekyoung Kwon
ktk@emerald.yonsei.ac.kr

Abstract

This paper presents a new password authentication and key agreement protocol called AMP in a provable manner. The intrinsic problem with password authentication is a password, associated with each user, has low entropy so that (1) the password is hard to transmit securely over an insecure channel and (2) the password file is hard to protect. Our solution to this complex problem is the amplified password proof idea along with the amplified password file. A party commits the high entropy information and amplifies her password with that information in the amplified password proof. She never shows any information except that she knows it for her proof. Our amplified password proof idea is similar to the zero-knowledge proof in that sense. A server stores amplified verifiers in the amplified password file that is secure against a server file compromise and a dictionary attack. AMP mainly provides the password-verifier based authentication and the Diffie-Hellman based key agreement, securely and efficiently. AMP is simple and actually the most efficient protocol among the related protocols.

1. Introduction

Entity authentication is necessary for identifying the entities who are communicating over an insecure network. This function is usually combined with a key establishment scheme such as key transport or key agreement among the parties. For user authentication, three kinds of approaches exist; *knowledge-based authentication*, *token-based authentication*, and *biometric authentication*. Among them, the knowledge-based scheme is aimed for human memory ($\approx$ mind). Actually it is the most widely-used method due to such advantages as simplicity, convenience, adaptability, mobility, and less hardware requirement. It requires users only to remember and type in their knowledge called a password. Therefore, it is allowed for users to move conveniently without carrying hardware tokens. However, a complex problem with this password-only authentication is a mnemonic password has low entropy so that it is vulnerable to guessing attacks. The problem becomes more critical in an open distributed environment. A password file protection is another problem that makes this approach more unreliable, for example, if a password file is compromised, an adversary is able to impersonate a server or launch dictionary attacks.

PASSWORD PROTOCOLS. Since the first scheme called LGSN[24] was introduced in 1989, many protocols have been developed. Among them, EKE[7] was a landmark of certificate-free protocols. One variant named DH-EKE[7] introduced the password authentication and key agreement, and was "augmented" to A-EKE[8] that was the first verifier-based protocol to resist a password-file compromise and to accommodate *salt*[37]. GLNS[15] was enhanced from LGSN. Due to the inefficiency and constraints of older schemes, various modifications and improvements have followed. They include TH[36], AL[1], M-EKE[35], Gong[16], KS[20], SPEKE[18, 19], S3P[33], SRP[38], HK[17], GXY[21], and TLS adaptation[11]. However, some of them have been broken and some are still being cryptanalyzed[2, 14, 29, 9]. Most were inadequate

for security proof due to the ad-hoc methods of protecting passwords. In the mean time, OKE[25] introduced a provable approach and was followed by elegant work such as SNAPI[26], EKE2[5], AuthA[6], and PAK[10]. They show the provable approach in this area is getting matured.

A-EKE, B-SPEKE, SRP, GXY, SNAPI-X, AuthA, and PAK-X are classified as password-verifier based protocols[8, 19, 38, 21, 26, 6, 10]. They allow the asymmetric model in which a client possesses a password while a server stores its verifier rather than the password. Following A-EKE[8], B-SPEKE was augmented from SPEKE[18, 19]. SRP showed efficient work on a verifier and GXY was derived from it[38, 21]. SNAPI-X was augmented from SNAPI while PAK-X was enhanced from PAK[26, 10]. AuthA was derived from several previous protocols but enriched with provable security[6]. Recently a pseudorandom moduli scheme was proposed though it may be relatively inefficient[30]. However, even the verifier-based protocols allow dictionary attacks and server impersonation attacks if a server file is compromised. Currently the standardization on this field is being considered by IEEE P1363 group.

CONTRIBUTION. Our goal is to design a new protocol in a provable manner, which combines the following functions securely and efficiently.

- Password(-verifier) based authentication[8]
- Diffie-Hellman based key agreement[13]
- Password file protection

For achieving the goal, we propose two simple ideas (1) the *amplified password proof* that makes a user amplify her mnemonic password with a high entropy source and prove that she knows it, and (2) the *amplified password file* that makes a server store the amplified verifier for resisting a server file compromise. From the point of view, we name our protocol AMP that stands for "Authentication and key agreement via Memorable Password". We also present several variants of AMP and compare the efficiency of all verifier-based protocols in the end. Actually AMP is the most efficient protocol with plentiful functions, among the existing verifier-based protocols. Security proof of AMP is handled in the full paper version[22].

2. AMP Protocol Design

2.1. Preliminaries

AMP is typically the two party case so that we use $Alice$ and Bob for describing a client and a server, respectively. Eve indicates an adversary regardless of her passivity and activity. π and τ denote a password and salt, respectively. $\doteq$ 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. k is our *security parameter* long enough to prevent brute-force attacks. We set $l(k) \geq 2k$, $\omega(k) \leq \frac{2}{3}k$, and $t(k) \leq \frac{1}{3}k$ when we assume the length of k is around 80 bits. $h() : \{0,1\}^* \mapsto \{0,1\}^{l(k)}$ means a collision-free one-way hash function such as SHA-1 and RIPEMD-160. All hash functions are assumed to behave like random oracles for security proof[3]. Note that we abbreviate a modular notation "$\bmod p$" for convenience hereafter.

RANDOM ORACLE. We assume random oracles $h_i() : \{0,1\}^* \mapsto \{0,1\}^{l(k)}$ for $i \in [1,5]$. 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. For practical recoveries of random oracles in the real world, we define; $h_1(x) = h(00|x|00)$, $h_2(x) = h(01|x|01)$, $h_3(x) = h(01|x|10)$, $h_4(x) = h(10|x|10)$ and $h_5(x) = h(11|x|11)$ by following the constructions given in the Bellare and Rogaway's work[3]. $|$ denotes the concatenation.

NUMERICAL ASSUMPTION. The security of AMP relies on two familiar hard problems which are believed infeasible to solve in polynomial time. One is the *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 $x \in [0, p-2]$. The other is the *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 properties in a prime-order subgroup[28].

We assume that all numerical operations are on the cyclic group where it is hard to solve these problems. We consider the multiplicative group Z_p^* and actually use its prime-order subgroup Z_q. For the purpose, Bob chooses g that generates a prime-order subgroup Z_q where $p = qr + 1$. Note that a prime q must be sufficiently large ($> l(k)$) to resist Pohlig-Hellman decomposition and various index-calculus methods but can be much smaller than p[28, 31, 32]. It is easy to make g by $\alpha^{(p-1)/q}$ where α generates Z_p^*. Z_q is preferred for efficiency and for preventing small subgroup confinement more effectively. By confining all exponentiation to the large prime-order subgroup through g of Z_q, each party of the protocol is able to detect on-line attacks whenever a received exponential is confined to a small subgroup, for example, a square root attack[28]. We can use a *secure prime* modulus p such that $(p-1)/2q$ is also prime or each prime factor of $(p-1)/2q$ is larger than q, or a *safe prime* modulus p such that $p = 2q+1$[23]. We strongly recommend to use the secure prime modulus because it is relatively easier to find[23] and allows much smaller q, e.g., close to $l(k)$.

2.2. Our Idea

Our idea is simply to "amplify" the low entropy of passwords with a high entropy source to prevent dictionary attacks. The so-called amplified password is a time-variant parameter with high entropy while the mnemonic password is a time-invariant parameter with low entropy. Therefore, it is easy to prove the security of the amplified password based protocol in the random oracle model[22]. On the basis of this idea, we secure (1) the registration of the password, (2) the transmission of the password information between the communicating parties and, (3) the password file maintained by a server.

DEFINITIONS. We give useful definitions for describing our idea.

Definition 1 *A Password Proof defines: a party A who knows a low entropy secret called a password makes a counterpart B convinced that A is who knows the password.*

If A is a user while B is a server, then this definition deals with a remote user access in a distributed environment. We can consider two kinds of setup for the password proof. They are (1) a symmetric setup in which both A and B uses a password for proof and (2) an asymmetric setup in which A uses a password while B uses its verifier for proof. The asymmetric setup could benefit from salt for making it difficult for adversaries to compile a dictionary of likely passwords. The asymmetric setup gives better security than the symmetric setup because a client impersonation is infeasible even if a server file is compromised. As for the security of transmitting the password information, we can define two kinds of password proof.

Definition 2 *A Secure Password Proof defines: a party A successfully performs the Password Proof without revealing the information about the password itself.*

Actually after the R number of trials with different likely passwords, an adversary will be allowed the $\frac{1}{2^{\omega(k)}-R}$ probability of a successful participation because the password is a time-invariant parameter. The probability is negligible to reveal the password information because wrong participations will be counted and denied by the counterpart. So we say the secure password proof does not reveal any information about the password.

Definition 3 *An Insecure Password Proof defines: a party A successfully performs the Password Proof but fails the Secure Password Proof, or a party A successfully performs the Password Proof by showing all or partial information about the password that is not negligible.*

The insecure proof can be classified into the fully insecure password proof such as PAP(password only), the partially insecure password proof such as CHAP(challenge and handshake), and the cryptographically insecure password proof such as some cryptographic protocols[1, 29].

Definition 4 *An Amplified Password Proof defines:*

a party A who knows a password amplifies the password with a high entropy source and makes a counterpart B convinced that A is who knows the amplified password.

THE AMPLIFICATION. Our amplification idea is very simple, for example, $Alice$ proves her knowledge of a password π by giving $x + \pi \bmod q$ rather than π only, while x is the randomly-chosen high entropy information. For the purpose, a fresh x must be committed securely by $Alice$ prior to her proof in each session. ($x + \pi$ is not guessable at all whereas π is guessable, if x is kept securely.)

Definition 5 *The Amplified Password ϖ defines a value that only who knows π and x can make from $\mathcal{A}(\pi; x)$ where x is chosen randomly at Z_q and π is a mnemonic password chosen at $\{0,1\}^{\omega(k)}$ for an arbitrary amplification function $\mathcal{A}()$.*

Note ϖ is time-variant while π is time-invariant. We configure this idea as an amplified password proof.

THE AMPLIFIED PASSWORD PROOF. Assume $Alice$ knows π and Bob has g^π. The *amplified password proof* is basically composed of three steps: (1) the *initial commitment* step performs a secure commitment of the high entropy information, (2) the *challenge* step transmits a random challenge, (3) the *response* step performs a knowledge proof about the amplified password ϖ. We define three functions for each step; they are $\mathcal{G}_1()$ for initial commitment, $\mathcal{G}_2()$ for a challenge, and $\mathcal{H}()$ for a response.

Definition 6 *The Amplified Password Proof performs: $Alice$ who knows her password π randomly chooses a high entropy source x and securely commits it to Bob. Bob who knows g^π picks y at random and asks $Alice$ if she knows the password and the committed information. $Alice$ responds with the fact she knows the amplified password ϖ that includes the password and the committed information.*

Alice		Bob
$initial\ commitment$	$\xrightarrow{\mathcal{G}_1(x)}$	
	$\xleftarrow{\mathcal{G}_2(y)}$	$challenge$
$response$	$\xrightarrow{\mathcal{H}(\varpi)}$	

For secure commitment, $\mathcal{G}_1()$ should not reveal x even to Bob. So we set $\mathcal{G}_1() = g^x$ relying on the one-way property of the modular exponentiation $x \mapsto g^x$. While $\mathcal{G}_2()$ transmits a fresh challenge, $\mathcal{H}()$ must imply the fact that $Alice$ knows ϖ without revealing any information about ϖ, x and π. If we set $\mathcal{A}(\pi; x) = (x + \pi)^{-1} \bmod q$, then only who knows x and π can compute ϖ where $x + \pi$ is not known. So we set $\mathcal{G}_2() = g^{(x+\pi)y}$ to transmit a random challenge without revealing the information related to π, x, and y. Then we set g^y as verification information. $Alice$ can make it by computing that $(g^{(x+\pi)y})^\varpi = g^y$. Bob who knows $\mathcal{G}_1()$ as well as g^π, can make $\mathcal{G}_2()$ by computing $(g^x g^\pi)^y$. As a result, both parties can get g^y, the verification information, so we set $\mathcal{H}() = g^y$ or its hash value. Of course, they can make g^{xy} due to the Diffie-Hellman scheme. We can derive the following theorem that is easy to prove by assuming x is randomly chosen at Z_q. (*hint* : ϖ is not derivable from g^x, $g^{(x+\pi)y}$ and g^y even if g^π is compromised. π as well as a fresh x are necessary for computing $\mathcal{A}(\pi; x)$.)

Theorem 1 *The Amplified Password Proof is a Secure Password Proof.*

This means $Alice$ never shows the password itself for her proof, rather she proves the fact of knowing it. The amplified password proof idea is very similar to the zero-knowledge proof in that sense, but g^π must be kept securely because (1) the entropy of π is extremely low, and (2) g^π can be used for a client impersonation as well as a server impersonation (we discuss it later).

THE AMPLIFICATION AND KEY EXCHANGE. It is easy to add key exchange to the amplified password proof because we already utilized the Diffie-Hellman scheme. For key exchange, $Alice$ can derive a session key from g^{xy} and show she agrees on it. Bob is also able to run the same thing. A strong one-way hash function must be the best tool for this. For $Alice$ who wishes to agree on g^{xy}, we set $\mathcal{A}(\pi; x) = (x + \pi)^{-1} x \bmod q$. For mutual key confirmation as well as mutual explicit authentication, however, the protocol must be con-

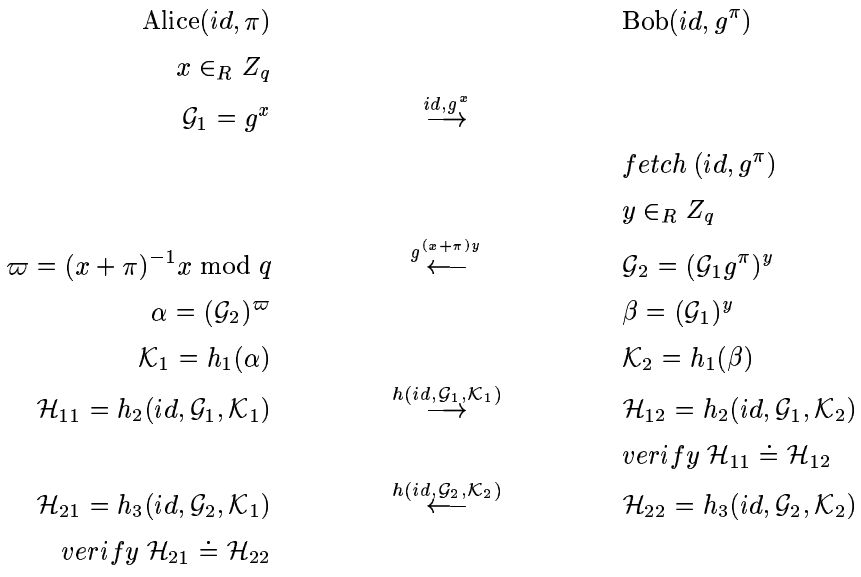

Figure 1. AMP^n Protocol

figured by four steps to add Bob's response. Figure 1 describes a basic version of our protocol. Note that the cases, $x \in \{0,1\}^1$, $y \in \{0,1\}^1$, $\mathcal{G}_1 \in \{0,1\}^1$, $\mathcal{G}_2 \in \{0,1\}^1$, and their small subgroup confinement must be avoided for a security reason. Both parties compute exponentials as like the Diffie-Hellman scheme. The difference is that a random exponent of α and a base of $\mathcal{G}_2$ are tactfully transformed. We call this protocol AMP$^{\mathsf{n}}$(AMP-naked) because it cannot provide the asymmetric setup security, i.e. it is vulnerable to a client impersonation if g^π is compromised. For example, if an adversary Eve who knows g^π sends $(g^\pi)^x$ to Bob, then Bob will respond with $\mathcal{G}_2 = (g^{\pi x} g^\pi)^y = g^{\pi y(x+1)}$ and compute $\beta = (g^{\pi x})^y$. Eve who chose x can cheat Bob by removing $x+1$ from $\mathcal{G}_2$ and raising it to x. As a result, AMP$^{\mathsf{n}}$ provides the security of the symmetric setup even if Bob stores g^π. So we can allow Bob to store π rather than g^π in this protocol. However, it is easy to preclude the client impersonation attack. Firstly, we propose an *e-protection* method for the purpose. If $Alice$ and Bob compute a time-variant parameter e, for example, $e = h(\mathcal{G}_1, \mathcal{G}_2, id, Alice, Bob)$, and embed it in α and β, then the password file is protected against the client impersonation attack (see section 3.1). There is more powerful idea named an amplified password file for improving the security of the password file.

THE AMPLIFIED PASSWORD FILE. As for the password file, an asymmetric setup is preferred because of the weakness of *text-equivalence* in a symmetric setup[8, 19, 38], meaning that the password file can be used for a client impersonation if it is compromised in the symmetric setup. However, the low entropy of passwords still makes the password file vulnerable to dictionary attacks and server impersonation attacks even if each password is hashed or exponentiated in the asymmetric setup, for example, a verifier such that $\upsilon = h(\pi)$. For the password file protection, encryption can be considered but key management and performance issues must be overcome. The amplified password file is a password file of which a record contains an amplified verifier for precluding all the related attacks.

Definition 7 *The Amplified Verifier ν defines a value that only who knows ς and τ can use for password verification where ς is chosen randomly at Z_q and τ is chosen randomly at $\{0,1\}^k$. Set $\nu = g^{(\varsigma+\tau)^{-1}\upsilon}$*

where $\upsilon = h(id, \pi)$. *If* $(\varsigma + \tau)^{-1} = 1$, ν *is not the amplified verifier. (Note:* ν *is semi-permanent.)*
A record of the amplified password file is (id, τ, ν). It is easy to update ς or τ in the amplified password file, e.g., by computing $\nu^{(\varsigma+\tau)(\varsigma'+\tau)^{-1}}$ where ς' is a new one. The amplified password file may be stored in a server storage but ς must be handled securely as a server's private key. It is recommended that ς should be loaded from a secure storage device such as a smart card when the system is initiated. Since ς resides in the server's run-time memory, a memory dump and its analysis are necessary for running a server impersonation attack or a dictionary attack with the compromised password file. It is easy to prove that the amplified password file is secure against such attacks if ς is kept securely.

Theorem 2 *The Amplified Password File is secure against password file compromise related attacks.*

AMP will be the protocol that enables those amplified password ideas.

3. AMP Protocol Family

This section describes AMP (Figure 2) and its variants in more detail.

3.1. AMP Protocol Description

We set $\mathcal{A}(\pi; x) = (x + \upsilon)^{-1}(x + e)$ where $\upsilon = h_1(id, \pi)$ and $e = h_2(\mathcal{G}_1, \mathcal{G}_2, id, Alice, Bob)$.

PROTOCOL SETUP. This step determines and publishes parameters of AMP.

1. Global Parameters: $Alice$ and Bob share g, p and q in an authentic manner. For example, Bob signs and publishes those parameters. (id indicates a precise user identifier while $Alice$ and Bob denote client and server entities respectively.)

2. Secure Registration: $Alice$ (or a user) chooses $\pi \in_R \{0,1\}^{\omega(k)}$ and notifies Bob in an authentic and confidential manner, for example, by either way of the following.

   (a) (on-line registration) $Alice$ computes g^{υ} where $\upsilon = h_1(id, \pi)$ and encrypts it along with a large random pad for precluding a *forward search attack* under Bob's public key. Otherwise, $Alice$ uses a *digital envelope* for encrypting g^{υ} under a random key. Then $Alice$ submits it to Bob.

   (b) (off-line registration) A user visits Bob's office and registers π with a picture id proof.

3. Server Storage: Bob chooses $\tau \in_R \{0,1\}^k$ and stores $(id, \tau, \nu = g^{(\varsigma+\tau)^{-1}\upsilon})$ after computing $(\varsigma+\tau)^{-1} \bmod q$ and $(g^{\upsilon})^{(\varsigma+\tau)^{-1}}$ under his private key ς. Bob should discard g^{υ} (and the raw data such as π or υ).

PROTOCOL RUN. Note that the cases, $x \in \{0,1\}^1$, $y \in \{0,1\}^1$, $\mathcal{G}_1 \in \{0,1\}^1$, $\mathcal{G}_2 \in \{0,1\}^1$, $\nu \in \{0,1\}^1$, and their small subgroup confinement must be avoided for a security reason. The following steps explain how the protocol is executed in Figure 2.

1. $Alice$ computes $\mathcal{G}_1 = g^x$ by choosing $x \in_R Z_q$ and sends $(id, \mathcal{G}_1)$ to Bob.

2. After receiving message 1, Bob loads τ and ν, and computes $\mathcal{G}_2 = (\mathcal{G}_1)^y \nu^{(\varsigma+\tau)y}$ by choosing $y \in_R Z_q$. This can be done efficiently by the simultaneous multiple exponentiation method[27]. Note that $\mathcal{G}_2 = (\mathcal{G}_1)^y \nu^{(\varsigma+\tau)y} = (g^x g^{\upsilon})^y$. He sends $\mathcal{G}_2$ to $Alice$.

3. While waiting for message 2, $Alice$ computes $\upsilon = h_1(id, \pi)$ and $\chi = (x + \upsilon)^{-1} \bmod q$. After receiving message 2, $Alice$ computes $e = h_2(\mathcal{G}_1, \mathcal{G}_2, id, Alice, Bob)$, $\varpi = \chi(x + e) \bmod q$ and $\alpha = (\mathcal{G}_2)^{\varpi}$. Note that $\alpha = (g^{(x+\upsilon)y})^{\varpi} = g^{y(x+e)}$. She computes $\mathcal{K}_1 = h_3(\alpha)$ and $\mathcal{H}_{11} = h_4(id, \mathcal{G}_1, \mathcal{K}_1)$. She sends Bob $\mathcal{H}_{11}$.

4. While waiting for message 3, Bob computes $e = h_2(\mathcal{G}_1, \mathcal{G}_2, id, Alice, Bob)$, $\beta = (\mathcal{G}_1)^y g^{ey} = (g^x g^e)^y = g^{(x+e)y}$, $\mathcal{K}_2 = h_3(\beta)$ and $\mathcal{H}_{12} = h_4(id, \mathcal{G}_1, \mathcal{K}_2)$. After receiving message 3, Bob compares $\mathcal{H}_{12}$ with $\mathcal{H}_{11}$. If they match, he computes $\mathcal{H}_{22} = h_5(id, \mathcal{G}_2, \mathcal{K}_2)$ and sends $Alice$ $\mathcal{H}_{22}$. This means he authenticated $Alice$ who knows ϖ (actually υ and thus

Alice (id, π)		Bob (id, τ, ν)
$x \in_R Z_q$		
$\mathcal{G}_1 = g^x$	$\xrightarrow{id, g^x}$	
		$fetch\ (id, \tau, \nu)$
$\upsilon = h_1(id, \pi)$		$y \in_R Z_q$
$\chi = (x + \upsilon)^{-1} \bmod q$	$\xleftarrow{g^{(x+\upsilon)y}}$	$\mathcal{G}_2 = (\mathcal{G}_1)^y \nu^{(\varsigma+\tau)y} = (\mathcal{G}_1 g^{\upsilon})^y$
$e = h_2(\mathcal{G}_1, \mathcal{G}_2, id, Alice, Bob)$		$e = h_2(\mathcal{G}_1, \mathcal{G}_2, id, Alice, Bob)$
$\varpi = \chi(x + e) \bmod q$		
$\alpha = (\mathcal{G}_2)^{\varpi}$		$\beta = (\mathcal{G}_1)^y g^{ey} = (\mathcal{G}_1 g^e)^y$
$\mathcal{K}_1 = h_3(\alpha)$		$\mathcal{K}_2 = h_3(\beta)$
$\mathcal{H}_{11} = h_4(id, \mathcal{G}_1, \mathcal{K}_1)$	$\xrightarrow{h(id, \mathcal{G}_1, \mathcal{K}_1)}$	$\mathcal{H}_{12} = h_4(id, \mathcal{G}_1, \mathcal{K}_2)$
		$verify\ \mathcal{H}_{11} \doteq \mathcal{H}_{12}$
$\mathcal{H}_{21} = h_5(id, \mathcal{G}_2, \mathcal{K}_1)$	$\xleftarrow{h(id, \mathcal{G}_2, \mathcal{K}_2)}$	$\mathcal{H}_{22} = h_5(id, \mathcal{G}_2, \mathcal{K}_2)$
$verify\ \mathcal{H}_{21} \doteq \mathcal{H}_{22}$		

Figure 2. AMP Protocol

π since x is secure from g^x), and agreed upon $\mathcal{K}(= \mathcal{K}_1 = \mathcal{K}_2)$.

5. While waiting for message 4, $Alice$ computes $\mathcal{H}_{21} = h_5(id, \mathcal{G}_2, \mathcal{K}_1)$. After receiving message 4, she compares $\mathcal{H}_{21}$ with $\mathcal{H}_{22}$. If they match, $Alice$ also agrees on $\mathcal{K}(= \mathcal{K}_1 = \mathcal{K}_2)$ with authenticating Bob who knows ν.

DISCUSSIONS. AMP passes four messages between $Alice$ and Bob who agree on $g^{(x+e)y}$ and explicitly authenticate each other while they agreed on g^{xy} in AMP$^{\mathsf{n}}$. In the full paper version[22], we give our security proof of AMP in the random oracle model derived from the Bellare and Rogaway's work[3, 4]. For security proof, we define a *Long-lived Weak-key generator* $\mathcal{W}()$ for π and a *Short-lived Strong-key generator* $\mathcal{S}()$ for ϖ with classifying sessions depending on the ability of the adversary in the random oracle model. We say $\mathcal{W}() \mapsto \mathcal{S}()$ due to $\mathcal{A}(\pi; x)$. Then we can prove the following theorem[22].

Theorem 3 AMP *is a secure authenticated key exchange protocol with* $\mathcal{W}()$.

An adversary Eve may need a memory dump and its analysis for getting ς even if she compromised a password file. The *e-protection* is necessary for the case that both the password file and ς are compromised. So Eve cannot falsely convince Bob that she is $Alice$ even with ς as well as the password file. $\mathcal{G}_2$ and $\alpha(= \beta)$ has a similar structure and their statistical difference relies on that of υ and e. However, it is not a critical point due to the beneficial property of random oracles. $Alice$ does not need to show π or υ even to Bob in secure on-line registration. Bob is able to update ς or τ in the amplified password file, for example, by computing $\nu^{(\varsigma+\tau)(\varsigma'+\tau)^{-1}}$ where ς' is a new private key. Final two steps can be modified, for example, $\mathcal{H}_{11} = h_4(\alpha, \mathcal{G}_1, \mathcal{G}_2, id, Alice, Bob)$ and $\mathcal{H}_{22} = h_5(\beta, \mathcal{G}_2, \mathcal{G}_1, Bob, Alice, id)$. We can choose salt τ implicitly by computing $f(id, B)$ where $f()$ is an implicit salt function[6, 10], for example, $\upsilon = h_1(id, f(id, B), \pi)$. For updating the existing system such as Unix, we can modify υ such that $\upsilon = h(\tau', \pi)$ and sends τ' in message 2 where $h(\tau', \pi)$ is an existing verifier for $\tau' \in_R \{0, 1\}^{t(k)}$.

3.2. AMP Protocol Variants

It is possible to derive variants from AMP and AMP^n for several issues. Here we summarize them briefly due to the page restriction.

AMP^a. This protocol is a variant that excludes the e-protection scheme from AMP. $Alice$ and Bob do not need to compute e for obtaining α or β respectively. Rather they agree on g^{xy} as we did in AMP^n. Note AMP^a is secure against a client impersonation even if the password file is compromised. This is due to the amplified password file only if Bob's private key is securely maintained. We set $\mathcal{A}(\pi; x) = (x + \upsilon)^{-1} x \bmod q$ for AMP^a.

AMP^e. This protocol is a variant that excludes the amplified password file from AMP. The security of the password file is only dependent upon the e-protection scheme so that a client impersonation is prevented but a server impersonation and a dictionary attack is possible if the password file is compromised. The difference in protocol setup is that Bob stores $(id, \tau, \nu = g^{\upsilon})$ where $\upsilon = h_1(\tau, \pi)$ and $\tau \in_R \{0,1\}^{t(k)}$. Of course, implicit salt can be used. The difference in protocol run is that $Alice$ should compute the amplified password after receiving $\mathcal{G}_2$. AMP^e does not need a simultaneous exponentiation method for $\mathcal{G}_2$ and a secure handling of Bob's private key, but loses the AMP level security against a password file compromise. We set $\mathcal{A}(\pi; x) = (x + \upsilon)^{-1}(x + e) \bmod q$ for AMP^e.

AMP^i. This protocol is a variant that allows "implicit authentication" for efficiency. If explicit authentication is not necessary, we can use AMP^i in which the parties are implicitly authenticated by using only first two steps of AMP (of course, it can be derived from AMP^a or AMP^e). Note that implicit authentication always requires a confidential session to be established. $Alice$ sends $\mathcal{G}_1$ to Bob who will respond with $\mathcal{G}_2$. Then, they can simply communicate with each other under the obtained session key. If one of them is not who is claimed to be, they cannot communicate because the dishonest party is not able to get the key.

AMP$^+$. This protocol is a variant that perturbs the structural similarity between $\mathcal{G}_2 = g^{(x+\upsilon)y}$ and $\alpha = \beta = g^{(x+e)y}$. However, such a similarity is not an issue at all due to the property of random oracles so that AMP$^+$ is a redundant protocol of AMP family. We give this protocol as a reference only. The main difference in protocol run is that both parties compute $e_1 = h_2(\mathcal{G}_1, id, Alice, Bob)$ and $e_2 = h_3(\mathcal{G}_1, \mathcal{G}_2, id, Alice, Bob)$. We set $\varpi = (xe_1 + \upsilon)^{-1}(x + e_2) \bmod q$ and $\mathcal{G}_2 = (\mathcal{G}_1^{e_1} g^{\upsilon})^y$. Note that $\mathcal{G}_2 = g^{(xe_1+\upsilon)y}$ while $\alpha = g^{(x+e_2)y}$.

AMP^{++}. This protocol is another form of a redundant protocol. The difference in protocol setup is that Bob stores $(id, \nu = g^{-(\varsigma+\tau)^{-1}\upsilon})$. Protocol run is different that $Alice$ chooses two ephemeral parameters such as x_1 and x_2. $Alice$ computes $\mathcal{G}_0 = x_1 + \upsilon \bmod q$ and $\mathcal{G}_1 = g^{x_2}$, sends them to Bob who will respond with $\mathcal{G}_2$. We set $\mathcal{A}(\pi; x) = (x_2 - \upsilon)^{-1}(x_1 + ex_2) \bmod q$ and $\mathcal{G}_2 = (\mathcal{G}_1^{e_1} g^{-\upsilon})^y$. Bob gets β by computing $(g)^{\mathcal{G}_0 y}(\nu)^{(\varsigma+\tau)y}(\mathcal{G}_1)^{ey}$. Therefore, the agreed key is $g^{(x_1+ex_2)y}$ while $\mathcal{G}_2 = g^{(x_2-\upsilon)y}$.

AMP^{n+}. AMP^n provided the symmetric setup security even if Bob stored a verifier g^{π}. We can extend this protocol for verifier-based authentication in the asymmetric setup model. The main difference in protocol setup is that $\nu = h_2(\tau, \upsilon)$ where $\upsilon = h_1(id, \pi)$ and $\tau \in_R \{0,1\}^{t(k)}$. Therefore, Bob can save a storage for ν compared to AMP, but loses several security benefits. We define functions such that $\mathcal{E}(x, y) = x + y \bmod q$ and $\mathcal{D}(x, y) = x - y \bmod q$, and set $\mathcal{H}_{11} = \mathcal{E}(e, \upsilon)$ and $\mathcal{H}_{22} = h_5(\mathcal{G}_2, \mathcal{K}_2)$ where $e = h_4(id, Alice, Bob, \mathcal{K}, \alpha)$. We set $\mathcal{A}(\pi; x) = (x + \nu)^{-1} x \bmod q$. When Bob receives $\mathcal{H}_{11}$, he can verify it by computing $h_2(\tau, \mathcal{D}(\mathcal{H}_{11}, e))$. We can replace the operations of $\mathcal{E}()$ and $\mathcal{D}()$ with a modular multiplication or a conventional encryption function. However,

AMP^{n+} loses the zero-knowledge property because Bob is always able to read υ in a protocol run. In addition, the protocol is vulnerable to dictionary attacks if α, β, or a password file is compromised.

4. Analysis and Comparison

4.1. Security of AMP

Following the security proof of AMP in the random oracle model[22], we summarize the security of AMP.

AMP provides *perfect forward secrecy* because the security of AMP relies on the Diffie-Hellman problem and the discrete logarithm problem. Even if π (or υ) is compromised, Eve cannot find old session keys because she is not able to solve the hard problems on $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{H}_1$ and $\mathcal{H}_2$. Note that the amplified password ϖ is time-variant due to the discrete logarithm problem, i.e., Eve must find x from $\mathcal{G}_1(= g^x)$ to recompose ϖ even if she knows π.

Denning-Sacco attacks(or stolen key attacks) are the case that Eve, who compromised an old session key, attempts to find π or to make the oracle accept her[12]. For the purpose, Eve has to solve the discrete logarithm problem to make a new amplified password even if an old $g^{(x+e)y}(= \alpha = \beta)$ has been compromised. It is also infeasible to check the difference between e and υ in $g^{(x+e)y}$ and $g^{(x+\upsilon)y}$ without solving the discrete logarithm of g^x. Therefore, AMP is secure against this attack.

Replay attacks are 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 in the corresponding session. The amplified password ϖ is also time-variant. Finding those parameters corresponds 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.

Small subgroup confinement such as a square root attack is defeated and avoided by confining the exponentials to the large prime-order subgroup. Intentional small subgroup confinement to Z_2 can be detected easily due to the strong property of a safe prime or a secure prime modulus.

On-line guessing attacks are detectable and the following off-line analysis can be frustrated, even if Eve attempts to disguise parties. Actually, Eve is able to perform the on-line attack to either party but its failure is countable. *Impersonation* of the party or a *man-in-the-middle attack* is also infeasible without knowing υ or $\nu^{(\varsigma+\tau)}$.

Off-line guessing attacks are also infeasible because Eve cannot analyze $\mathcal{G}_2$. *Partition attacks* are to reduce the set of likely passwords logarithmically by asking the oracle in parallel with off-line analysis, while *chosen exponent attacks* are to analyze it via her chosen exponent. Both attacks are infeasible because Eve cannot solve or reduce $y' = (x+\upsilon)y(x+\upsilon')^{-1} \bmod q$ for guessed passwords without knowing both x and y.

Security against *password-file compromise* is the basic property of AMP family except AMP^n that has a naked property. Among them, AMP, AMP^a, AMP^i, AMP$^+$, and AMP^{++} provides the stronger security without degrading performance through the amplified password file.

4.2. Efficiency and Constraints

We examine the efficiency of AMP and compare it with other related protocols.

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 $\mathcal{G}_2$. For AMP^{++}, the size of $\mathcal{G}_0$ can be bounded by $l(k) + \epsilon$ with a negligible ϵ when we use a secure prime modulus.

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 $E(Alice : Bob)$. AMP has only $3E$ so that the best performance is expected. AMP has $E(g^x : -)$, $E(- : (\mathcal{G}_1)^y \nu^{(\varsigma+\tau)y})$ and $E(\mathcal{G}_2^{\varpi} : \mathcal{G}_1^y g^{ey})$ while all variants have similar operations. Here '$-$' means there is no modular exponentiation needing $O((\log n)^3)$. Note that AMP operations should benefit from the simultaneous multiple exponentiation method for efficiency[34, 27]. As for

	Protocol Steps	*Large Blocks*	*Exponentiations*			*Random Numbers*	
			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)	**2** (+0)	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	5 (+2) / **3** (+0)	**2** (+0)	4 (+2)	3 (+1)	6 (+3)	**1** (+0)	**1** (+0)
PAK-X	5 (+2) / **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. Comparison of Verifier-based Protocols

$g_1{}^{e_1} g_2{}^{e_2}$, we don't need to compute $g_1{}^{e_1}$ and $g_2{}^{e_2}$ separately. A simple description of the simultaneous method is as follows;

```
t = length(e);
g_x = g_1 g_2 mod p;
G[0]=1; G[1]=g_1; G[2]=g_2; G[3]=g_x;
A = 1;
for(i=1;i<=t;i++)
  {B_i = ExponentArray(i);}
for(i=1;i<=t;i++)
  {A = A*A mod p; A=A*G[B_i] mod p;}
return(A);
```

This scheme computes $g_1{}^{e_1} g_2{}^{e_2}$ by performing $t-1$ squarings and at most $t+1$ multiplications where each exponent is represented by t bits[34, 27].

Each party of AMP performs only two exponentiations regarding the efficiency of the simultaneous multiple exponentiation.

For run time parameters, each party generates one random number in AMP family except for AMP^{++}. $Alice$ can reduce her run time exponentiations to only once and parallel exponentiations to only twice by pre-computation of g^x.

In step 3, $Alice$ should compute $(x+v)^{-1}$ but only in the q-order subgroup. Modular inversion, $O((\log q)^2)$, is less expensive than modular exponentiation, $O((\log p)^3)$. Moreover, the size of q can be bounded by $l(k)+\epsilon$ with a negligible ϵ when we use a secure prime modulus. Note that $O(\log l(k)) << O(\log p)$.

AMP can be implemented on the elliptic curve group. A generalization on the elliptic curve group gives further efficiency, e.g., the size of a message and a password file.

Efficiency can be compared to the other related protocols such as A-EKE, B-SPEKE, SRP, GXY, SNAPI-X, AuthA and PAK-X[8, 19, 38, 21, 26, 6, 10]. Table 1 compares them in terms of the number of protocol steps, large message blocks, and exponentiations. SRP can benefit from the simultaneous exponentiation method only for a server side. Note that AuthA and PAK-X have five steps with explicit salt and three steps with implicit salt. The number of random numbers is given as a subsidiary reference. The number of parallel exponentiations could compare approximately the amount of protocol execution time. The value in parenthesis implies the difference from the most efficient one that is denoted by bold characters. Note that AMP provides the stronger security against the password file compromise compared to all the others in Table 1.

CONSTRAINT. We recommend to use a large ($> l(k)$) prime-order subgroup Z_q for defeating and avoiding the small subgroup confinement effectively by confining exponentials into the large prime-order subgroup[28]. A secure prime modulus is highly recommended for further efficiency of the protocols. Note that the secure prime is easier to get than the safe prime[23]. 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. Why AMP

We summarize various advantages of AMP.

1. AMP is a secure password(-verifier) based protocol equipped with the amplified password proof and the amplified password file. The security of AMP is proved in the random oracle model.
2. AMP is the most efficient protocol among the existing verifier-based protocols. AMP provides the best efficiency even with the amplified password file.
3. AMP has light constraints and is easy to generalize, e.g., in elliptic curve groups for further efficiency.
4. AMP has several variants for flexibility.
5. AMP allows the Diffie-Hellman based key agreement.
6. AMP has a simple structure so that it is easy to understand and implement the protocol.
7. AMP provides an easy way to upgrade the existing system. AMP accommodates any kinds of salt schemes without degrading performance.

5. Conclusion

In this paper, we introduced a new protocol called AMP and its variants for password authentication and key agreement. AMP has been designed on the basis of the amplified password proof and the amplified password file ideas. A time-variant parameter called the amplified password makes the protocol simple and easy to prove in the random oracle model[22]. Many password-based solutions such as Telnet, FTP, RADIUS and Kerberos are vulnerable to dictionary attacks[39]. AMP can be used to improve their security in an open distributed environment.

Acknowledgment The authors thank David Wagner, Doug Tygar, David Jablon, Tom Wu, Radia Perlman, Li Gong and anonymous reviewers for their helpful comments and kind suggestion to this work. We also thank Jooseok Song and Chitoos Ramamoorthy for their kind concern.

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," In *Asiacrypt 96*, 1996

[3] M. Bellare and P. Rogaway, "Entity authentication and key distribution," In *CRYPTO 93*, 1993

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

[5] M. Bellare, D. Pointcheval and P. Rogaway, "Authenticated key exchange secure against dictionary attack," In *Eurocrypt 00*, 2000

[6] M. Bellare and P. Rogaway, "The AuthA protocol for password-based authenticated key exchange," 2000, available from `http://www.cs.ucdavis.edu/ rogaway/papers/autha.ps`

[7] 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

[8] 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

[9] M. Boyarsky, "Public-key cryptography and password protocols: the multi-user case," In *ACM Conference on Computer and Communications Security*, 1999

[10] V. Boyko, P. MacKenzie and S. Patel, "Provably secure password authenticated key exchange using Diffie-Hellman," In *Eurocrypt 00*, 2000

[11] P. Buhler, T. Eirich, M. Steiner and M. Waidner, "Secure Password-Based Cipher Suite for TLS," In *Network and Distributed System Security Symposium*, February 2-4, 2000

[12] D. Denning and G. Sacco, "Timestamps in key distribution protocols," *Communications of the ACM*, vol.24, no.8, pp.533-536, 1981

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

[14] Y. Ding and P. Hoster, "Undetectable on-line password guessing attacks," *ACM Operating Systems Review*, vol.29, no.4, pp.77-86, October 1995

[15] L. Gong, M. Lomas, R. Needham, and J. Saltzer, "Protecting poorly chosen secrets from guessing attacks," *IEEE Journal on Selected Areas in Communications*, vol.11, no.5, pp.648-656, June 1993

[16] L. Gong, "Optimal authentication protocols resistant to password guessing attacks," In *IEEE Computer Security Foundation Workshop*, pp. 24-29, June 1995

[17] S. Halevi and H. Krawczyk, "Public-key cryptography and password protocols," In *ACM Conference on Computer and Communications Security*, 1998

[18] D. Jablon, "Strong password-only authenticated key exchange," *ACM Computer Communications Review*, vol.26, no.5, pp.5-26, 1996

[19] D. Jablon, "Extended password key exchange protocols," In *WETICE Workshop on Enterprise Security*, pp.248-255, 1997

[20] 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

[21] 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

[22] T. Kwon, "Authentication and key agreement via memorable password," 2000, available from `http://eprint.iacr.org/2000/026`

[23] C. Lim and P. Lee, "A key recovery attack on discrete log-based schemes using a prime order subgroup," In *CRYPTO*, pp.249-263, 1997

[24] M. Lomas, L. Gong, J. Saltzer, and R. Needham, "Reducing risks from poorly chosen keys," *ACM Symposium on Operating System Principles*, pp.14-18, 1989

[25] S. Lucks, "Open key exchange: how to defeat dictionary attacks without encrypting public keys," *The Workshop on Security Protocols*, April 7-9, 1997

[26] P. MacKenzie and R. Swaminathan, "Secure network authentication with password identification," 1999, available from `http://grouper.ieee.org/groups/1363/StudyGroup/Passwd.html#MS`

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

[30] R. Perlman and C. Kaufman, "Strong Password-Based Authentication Using Pseudorandom Moduli," IETF Internet Draft, 2000, available from `http://search.ietf.org/internet-drafts/draft-perlman-strong-pass-00.txt`

[31] 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

[32] J. Pollard, "Monte Carlo methods for index computation $\mathrm{mod}\ p$," *Mathematics of Computation*, vol.32, pp.918-924, 1978

[33] M. Roe, B. Christianson and D. Wheeler, "Secure sessions from weak secrets," Technical report from University of Cambridge and University of Hertfordshire, 1998, available from `http://www.ccsr.cam.ac.uk/techreports/tr4/index.html`

[34] C. Schnorr, "Efficient identification and signatures for smart cards," In *CRYPTO 89*, pp.239-251, 1989

[35] 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

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

[37] V. Voydock and S. Kent, "Security mechanisms in high-level network protocols," *Computing Surveys*, vol.15, no.2, pp.135-171, June 1983

[38] T. Wu, "Secure remote password protocol," In *Network and Distributed System Security Symposium*, 1998

[39] T. Wu, "A Real-World Analysis of Kerberos Password Security," In *Network and Distributed System Security Symposium*, February 3-5, 1999