Computer Security Symposium 2013
21 - 23 October 2013

Augmented パスワード認証方式の Sanity Check について

辛　星漢 †　　　古原　和邦 †

† 産業技術総合研究所セキュアシステム研究部門
305-8568 茨城県つくば市梅園 1-1-1
seonghan.shin@aist.go.jp

あらまし 本稿では IEEE 1363.2 と ISO/IEC 11770-4 に標準化された Augmented パスワード認証方式の sanity checks について検討する。

On Augmented Password-Authenticated Key Exchange

SeongHan Shin†　　Kazukuni Kobara†

†Research Institute for Secure Systems, AIST
1-1-1 Umezono, Tsukuba City, Ibaraki 305-8568, JAPAN
seonghan.shin@aist.go.jp

Abstract In this document, we investigate APKAS-AMP in IEEE 1363.2 and KAM3 in ISO/IEC 11770-4 which require several validity checks on the values, received and computed by the parties, when using a secure prime.

1 Introduction

The Password-Authenticated Key Exchange (PAKE) protocols provide password-only authentication and establishment of temporal session keys to be used for subsequent cryptographic algorithms. These protocols are designed to be secure against passive/active attacks as well as off-line dictionary attacks on human-memorable passwords, shared by the participating parties. For a long time, PAKE protocols have received much attention because password authentication is commonly used and widely deployed in practice. Since the appearance of Encrypted Key Exchange (EKE) [1, 2], a number of PAKE protocols (see [10] and references therein) have been proposed in the literature. And, some PAKE protocols have been standardized in IEEE 1363.2 [7], ISO/IEC 11770-4 [8] and IETF [16].

In general, PAKE protocols can be classified into 'balanced' PAKE and 'augmented' PAKE [7, 8]: in the former case a client and a server share a common password; and in the latter case a client remembers his/her password and a server has password verification data (derived by applying a one-way function to the password). Since password verification data has the same entropy of the password, the off-line dictionary attacks are inevitable if server is compromised in the augmented PAKE protocols. Nonetheless, an augmented PAKE protocol may be preferable because it provides extra protection for server compromise. Actually, there has been a significant amount of works (e.g., [5, 4]) on making PAKE protocols secure even in the case of server compromise.

There is an exceptional augmented PAKE protocol (AMP and its variants [11, 12, 13, 14, 15]), not based on balanced one. Among AMP and its variants, AMP2 [13] and AMP$^+$ [12] have been standardized in IEEE 1363.2 [7] and ISO/IEC 11770-4 [8], respectively. Though AMP and its variants do not give any security proofs (i.e., reduction to a computationally-hard problem), IEEE P1363.2 working group has chosen AMP as an augmented PAKE protocol due to the computational efficiency on client side (see Annex C.2.5 of IEEE 1363.2 [7]).

1.1 Our Contributions

In this paper, we investigate APKAS-AMP (based on AMP2 [13] and standardized in IEEE 1363.2 [7]) and KAM3 (based on AMP$^+$ [12] and standardized in ISO/IEC 11770-4 [8]) which require several validity checks on the values, received and computed by the parties, when using a secure prime p. After showing some attacks on APKAS-AMP and KAM3, we suggest new sanity checks that are clear and sufficient to prevent an attacker from doing any possible attacks (to be discussed in this paper).

1.2 Notation

Here, we explain some notation to be used throughout this paper. Let $\mathbb{G}$ be a finite, cyclic subgroup of prime order q of the multiplicative group $\mathbb{Z}_p^\star$ where $p = aq + 1$ is a prime such that $(a/2)$ is also a prime [9]. Such p is called a secure prime. Let g be a generator of $\mathbb{G}$ in which the group operation is denoted multiplicatively. The (p, q, g) are public to everyone and (p, q) are called domain parameters. In the aftermath, all the subsequent arithmetic operations are performed in modulo p unless otherwise stated.

Let k be the security parameter for hash functions. Let $\{0,1\}^\star$ denote the set of finite binary strings and $\{0,1\}^k$ the set of binary strings of length k. We use two different hash functions $\mathcal{H}$ and $\overline{\mathcal{H}}$ where $\mathcal{H} : \{0,1\}^\star \rightarrow \mathbb{Z}_q^\star$ and $\overline{\mathcal{H}} : \{0,1\}^\star \rightarrow \{0,1\}^k$. Let KCF and KDF be the key confirmation function and key derivation function, respectively, both of which are instantiated with secure one-way hash functions $\overline{\mathcal{H}}$ (see Section 12.3 and 11 of [7]). Also, let $A \| B$ be the concatenation of bit strings of A and B in $\{0,1\}^\star$. Let C and S be the identities of client and server, respectively, with each identity ID $\in \{0,1\}^\star$.

2 APKAS-AMP

In this section, we describe APKAS-AMP (Section 9.5 of IEEE 1363.2 standard [7]) in detail where APKAS is an acronym for Augmented Password-Authenticated Key Agreement Scheme. For computational efficiency of APKAS-AMP, it is strongly recommended to use a secure prime p (defined in Section 1.2). The APKAS-AMP scheme is actually based on AMP2 [13], and it consists of registration phase and key agreement operation phase.

2.1 Registration

In the registration phase, client C registers his/her password verification data $V \equiv g^v$ securely to server S where $v = \mathcal{H}(pw)$ and pw is the client's password. After this phase, client C just remembers his/her password pw and server S stores password verification data V on its database. Note that this phase is done only once.

2.2 Key Agreement Operation

Whenever client C and server S need to share an authenticated session key SK, they execute the below key agreement operation over insecure networks. During the key agreement operation, the APKAS-AMP scheme requires several validity checks on the values, received and computed by the parties.

Step 1: The client C selects a random private key x from the range $[1, q-1]$[1] and computes its public key $X \equiv g^x$. Then, client C sends the first message (C, X) to server S.

$$C \to S : (C, X)$$

Step 2: After receiving (C, X), server S checks whether the client's public key X is in the parent group or not. If $X \notin [1, p-1]$, it outputs "invalid" and stops (Server validation 1). Otherwise, server S selects a random private key y from the range $[1, q-1]$ and computes a password-entangled public key $Y \equiv (X^u \cdot V)^y$ where $u = \mathcal{H}(X \| C \| S)$ and V is the client's password verification data. Then, server S sends the second message (S, Y) to client C.

$$S \to C : (S, Y)$$

Step 3: After receiving (S, Y), client C checks whether the server's public key Y is in a specific range or not. If $Y \notin [2, p-2]$, it outputs "invalid" and stops (Client validation 1). Otherwise, client C computes a shared secret key $Z_C \equiv Y^{(x+1)/(x \cdot u + v)}$, where $u = \mathcal{H}(X \| C \| S)$ and $v = \mathcal{H}(pw)$, and an authenticator $V_C = \mathsf{KCF}(1, X, Y, Z_C)$. Then, client C sends the third message V_C to server S.

$$C \to S : V_C$$

Step 4: The server S computes a shared secret key $Z_S \equiv (X \cdot g)^y$. If the order of Z_S is unacceptably small, it outputs "invalid" and stops (Server validation 2, this will be explained in Section 3.2). If $V_C \neq \mathsf{KCF}(1, X, Y, Z_S)$, it outputs "invalid" and stops (Server validation 3). Otherwise, server S computes an authenticator $V_S = \mathsf{KCF}(2, X, Y, Z_S)$ and a session key $SK = \mathsf{KDF}(Z_S)$. Then, server S sends the fourth message V_S to client C.

$$S \to C : V_S$$

Step 5: The client C receives the authenticator V_S. If $V_S \neq \mathsf{KCF}(2, X, Y, Z_C)$, it outputs "invalid" and stops (Client validation 2). Otherwise, client C computes a session key $SK = \mathsf{KDF}(Z_C)$.

As in **Step 4**, server S should first confirm the client's proof of knowledge of shared secret key Z because the client does not provide a commitment to the password during the key agreement operation.

3 A New Sanity Check for APKAS-AMP

Here, we suggest a new sanity check for the APKAS-AMP scheme. This sanity check is clear and sufficient to prevent an attacker from doing any possible attacks (to be discussed in Section 3.1 and 3.2).

3.1 When $X \equiv (p-1)$

If client C inadvertently sends a public key $X \equiv (p-1)$ to server S, a passive attacker $\mathcal{M}$ can perform off-line dictionary attacks on the communication messages $(X, Y, V_{C/S})$. When $X \equiv (p-1)$, a password-entangled public key Y (computed and sent by server S) and a shared secret key $Z_{C/S}$ (included in an authenticator $V_{C/S}$) will be as follows:

$$\begin{aligned} Y &\equiv ((p-1)^u \cdot V)^y \equiv (-1)^{u \cdot y} \cdot (g^v)^y \\ &\equiv \left((-1)^{\frac{u \cdot y}{v}} \cdot g^y\right)^v \end{aligned}$$

and

$$Z_{C/S} \equiv ((p-1) \cdot g)^y \equiv (-1)^y \cdot g^y \ .$$

[1]Refer to D.5.2.1 of IEEE 1363a-2004 [6] for selecting a private key from a uniformly random distribution over the full range of private keys.

Note that $X \equiv (p-1)$ is a valid value in the check of Server validation 1. Also, it is clear that $X \equiv (p-1)$ exists since the discrete logarithm of X (i.e., $x \equiv \log_g(p-1) \equiv \log_g(aq) \bmod q$) is in the range $[1, q-1]$.

Because the server's private key y is randomly selected, the attacker $\mathcal{M}$ can test if $V_C \stackrel{?}{=} \mathsf{KCF}(1, X, Y, \pm Y^{1/v'})$ for all possible values (i.e., passwords) $v' = \mathcal{H}(pw')$. After these tests, attacker $\mathcal{M}$ finally finds out the correct client's password pw. Though the probability of $X \equiv (p-1)$ is negligible in the security parameter for $\mathbb{G}$, this attack might be meaningful because it is possible without knowing the discrete logarithm of X.

A countermeasure to the above attack is clear in the check of Server validation 1: If $X \notin [1, p-2]$, it outputs "invalid" and stops. Note that $X \equiv 1$ does not exist in the passive attack, and any active attacks are not possible when $X \equiv 1$.

3.2 Server Validation 2

In **Step 4** of the APKAS-AMP scheme, server S checks if the order of Z_S is unacceptably small or not (Server validation 2). For that, IEEE 1363.2 standard [7] explains how and why one should validate that Z_S is not a small order group element and the meaning of "unacceptably small". From Appendix D.2.2.1.4 of [7],

> "Without this check, an attacker impersonating client C could choose a small order element e, send $X \equiv e/g$ to server S, and confine Z_S to a small group, and thus determine the server's value for Z_S without knowledge of pw."

and, for the meaning of "unacceptably small", from Appendix D.2.1.5 of [7]

> "Some schemes include steps for verifying that the order of a group element e is not "unacceptably small" in order to defend against small subgroup attacks. In a small subgroup confinement attack, an attacker selects e, modifies e, or causes a party to compute e so that it generates an unacceptably small group, in an attempt to confine a subsequently-derived secret value Z to a set that is enumerable by the attacker.∼ However, it may be sufficient to merely ensure that e is a generator of any group of b or more elements, where the implementation determines security parameter b such that any e of order less than b is rejected. This ensures that an attacker cannot confine the legitimate party's derived secret Z to a set with less than b elements, thus ensuring that the attacker using a random password pw has no greater than a $1/b$ probability of negotiating shared key Z in each run.[2]∼ One way to simplify addressing small subgroup confinement is to choose domain parameters such that there are no non-trivial factors of $(p-1)/2$ smaller than q."

Now, we clarify the check of Server validation 2 from the well-known number theory facts when p is a secure prime.

Fact 1 (Euler's Theorem) *Let n be a positive integer and $\alpha \in \mathbb{Z}_n^\star$. Then, $\alpha^{\varphi(n)} \equiv 1$ where $\varphi(\cdot)$ is Euler's phi function. In particular, the multiplicative order of α divides $\varphi(n)$.*

As a consequence of Fact 1, we obtain Fermat't little theorem that, for every prime p and every $\alpha \in \mathbb{Z}_p$, $\alpha^p \equiv \alpha$.

[2] Essentially, b indicates a desired level of resistance to on-line dictionary attacks [7].

Fact 2 *Let p be an odd prime and $\beta \in \mathbb{Z}_p$. Then, $\beta^2 \equiv 1$ if and only if $\beta \equiv \pm 1$.*

Proof. It is obvious that, if $\beta \equiv \pm 1$, then $\beta^2 \equiv 1$. Conversely, suppose that $\beta^2 \equiv 1 \pmod p$, which means that

$$p \mid (\beta^2 - 1) = (\beta - 1)(\beta + 1) \ .$$

Since p is prime, we must have $p \mid (\beta - 1)$ or $p \mid (\beta + 1)$. This implies that $\beta \equiv \pm 1$. □
This fact says that the only square roots of 1 are $\pm 1 \pmod p$, which obviously belong to distinct residue classes (since $p > 2$).

From Fact 1 and 2, we have the following theorem for Server validation 2:

Theorem 1 *Let $p = 2a'q + 1$ be a prime such that (a', q) are also primes and $a' > q > 2$. In Server validation 2, the order (smaller than q) of Z_S is 2 if and only if $X \equiv \pm g^{-1}$.*

Proof. By Fact 1 and Fermat's little theorem, the multiplicative order (smaller than q) of Z_S is 2 since $Z_S \not\equiv 0$, $\varphi(p) = 2a'q$ and $a' > q > 2$. So, we have

$$(Z_S)^2 \equiv 1 \Longleftrightarrow (X \cdot g)^2 \equiv 1 \Longleftrightarrow X \equiv \pm g^{-1}$$

where the last congruence is derived from Fact 2. In other words, the order (smaller than q) of Z_S is 2 if and only if $X \equiv \pm g^{-1}$. □
This theorem guarantees that the only unacceptably small order elements Z_S of the parent group $\mathbb{Z}_p^\star$ exist when $X \equiv \pm g^{-1}$. Note that any value Z_S, when $X \not\equiv \pm g^{-1}$ and $X \not\equiv 0$ (already excluded by Server validation 1), is either of order a' or order q.

With Theorem 1, we can simplify the check of Server validation 2 as follows: If $X \in \{\pm g^{-1}\}$, it outputs "invalid" and stops.

3.3 In Summary

By combining the results of Section 3.1 and 3.2, we have a new sanity check for the APKAS-AMP scheme.

Step 1': Same as **Step 1** of Section 2.2

Step 2': After receiving (C, X), server S checks whether the client's public key X is in a specific range or not. If $X \notin [1, p-2] \setminus \{\pm g^{-1}\}$, it outputs "invalid" and stops (Server validation 1). Otherwise, server S selects a random private key y from the range $[1, q-1]$ and computes a password-entangled public key $Y \equiv (X^u \cdot V)^y$ where $u = \mathcal{H}(X \| C \| S)$ and V is the client's password verification data. Then, server S sends the second message (S, Y) to client C.

$$S \rightarrow C : (S, Y)$$

Step 3': Same as **Step 3** of Section 2.2

Step 4': The server S computes a shared secret key $Z_S \equiv (X \cdot g)^y$. If $V_C \neq \mathsf{KCF}(1, X, Y, Z_S)$, it outputs "invalid" and stops (Server validation 3). Otherwise, server S computes an authenticator $V_S = \mathsf{KCF}(2, X, Y, Z_S)$ and a session key $SK = \mathsf{KDF}(Z_S)$. Then, server S sends the fourth message V_S to client C.

$$S \rightarrow C : V_S$$

Step 5': Same as **Step 5** of Section 2.2

In **Step 4'**, Server validation 2 is no longer needed. As said before, the check of $X \notin [1, p-2] \setminus \{\pm g^{-1}\}$ in Server validation 1 is sufficient to prevent any possible attacks, discussed in Section 3.1 and 3.2.

4 Key Agreement Mechanism 3

In this section, we describe Key Agreement Mechanism 3 (Section 6.3 of ISO/IEC 11770-4 [8]) which is different from the APKAS-AMP scheme in IEEE 1363.2 standard [7]. Note that ISO/IEC 11770-4 [8] restricts domain parameters to a secure prime $p = aq + 1$ satisfying co-factor $a = 2r_1 r_2 \cdots r_t$ for primes $r_i > q$, $i = 1, 2, \cdots, t$ (optionally, $t = 0$). The Key Agreement Mechanism 3 (for short, KAM3) is actually based on AMP$^+$ [12], and it consists of registration phase and key agreement operation phase.[3]

4.1 Registration

In the registration phase, client C registers his/her password verification data $V \equiv g^v$ securely to server S where $v = \mathcal{H}(pw)$ and pw is the client's password. After this phase, client C just remembers his/her password pw and server S stores password verification data V on its database. Note that this phase is done only once.

4.2 Key Agreement Operation

Whenever client C and server S need to share an authenticated session key SK, they execute the below key agreement operation over insecure networks. During the key agreement operation, the KAM3 requires several validity checks on the values, received and computed by the parties.

Step 1: The client C selects a random private key x from the range $[1, q-1]$ and computes its public key $X \equiv g^x$. Then, client C sends the first message (C, X) to server S.

$$C \rightarrow S : (C, X)$$

Step 2: After receiving (C, X), server S checks whether the client's public key X is in the parent group or not. If $X \notin [1, p-1]$, it outputs "invalid" and stops (<u>Server validation 1</u>). Otherwise, server S selects a random private key y from the range $[1, q-1]$ and computes a password-entangled public key $Y \equiv (X^{u_1} \cdot V)^y$ where $u_1 = \mathcal{H}(1\|X)$ and V is the client's password verification data. Also, server S checks whether the Y is in a specific range or not. If $Y \notin [2, p-2]$, it outputs "invalid" and stops (<u>Server validation 2</u>). Otherwise, server S sends the second message (S, Y) to client C.

$$S \rightarrow C : (S, Y)$$

Step 3: After receiving (S, Y), client C checks whether the server's public key Y is in the parent group or not. If $Y \notin [1, p-1]$, it outputs "invalid" and stops (<u>Client validation 1</u>). Otherwise, client C computes a shared secret key $Z_C \equiv Y^{(x+u_2)/(x \cdot u_1 + v)}$, where $u_1 = \mathcal{H}(1\|X)$, $u_2 = \mathcal{H}(2\|X\|Y)$ and $v = \mathcal{H}(pw)$, and an authenticator $V_C = \overline{\mathcal{H}}(1\|X\|Y\|Z_C)$. Then, client C sends the third message V_C to server S.

$$C \rightarrow S : V_C$$

Step 4: The server S computes a shared secret key $Z_S \equiv (X \cdot g^{u_2})^y$ where $u_2 = \mathcal{H}(2\|X\|Y)$. If $V_C \neq \overline{\mathcal{H}}(1\|X\|Y\|Z_S)$, it outputs "invalid" and stops (<u>Server validation 3</u>). Otherwise, server S computes an authenticator $V_S = \overline{\mathcal{H}}(2\|X\|Y\|Z_S)$ and a session key $SK = \mathsf{KDF}(Z_S)$. Then, server S sends the fourth message V_S to client C.

$$S \rightarrow C : V_S$$

Step 5: The client C receives the authenticator V_S. If $V_S \neq \overline{\mathcal{H}}(2\|X\|Y\|Z_C)$, it outputs "invalid" and stops (<u>Client validation 2</u>). Otherwise, client C computes a session key $SK = \mathsf{KDF}(Z_C)$.

Like the APKAS-AMP scheme in Section 2.2, server S should first confirm the client's proof of knowledge of shared secret key Z.

The main differences between KAM3 (in ISO/IEC 11770-4 [8]) and APKAS-AMP (in IEEE 1363.2 [7]) are in the computation of shared secret key $Z_{C/S}$ and some validity checks (i.e., Client validation 1 and Server validation 2).

[3] This AMP$^+$ [12] corresponds to AMP [13].

5 A New Sanity Check for KAM3

After showing an impersonation/passive attack on the KAM3, we suggest a new sanity check that is sufficient to prevent any possible attacks (to be discussed in Section 5.1 and 5.2).

5.1 An Impersonation Attack on KAM3

Here, we show a simple impersonation attack on KAM3 to break semantic security of session keys and server authentication (defined in [3]) with probability 1.

Suppose an attacker $\mathcal{M}$ who impersonates server S by sending a public key $Y \equiv \pm 1$. Note that $Y \equiv \pm 1$ is a valid value in the check of Client validation 1. In case of $Y \equiv 1$, attacker $\mathcal{M}$ can compute a correct authenticator V_S and session key SK since $Z_{C/S} \equiv 1$. In case of $Y \equiv (p-1)$, attacker $\mathcal{M}$ first finds out d, satisfying $V_C = \overline{\mathcal{H}}(1\|X\|Y\|d)$ for $d \equiv \pm 1$, and then can compute a correct authenticator V_S and session key SK with $Z_{C/S} \equiv d$. In both cases, the attacker $\mathcal{M}$ does not need to know the password pw.

A countermeasure to the above attack is clear in the check of Client validation 1: If $Y \notin [2, p-2]$, it outputs "invalid" and stops. At the same time, Server validation 2 in **Step 2** is no longer needed because $Y \equiv \pm 1$ (computed and sent by server S) does not pass the new validity check and does not give any information about the password pw.

5.2 When $X \equiv (p-1)$

Similar to Section 3.1, a passive attack when $X \equiv (p-1)$ is also applicable to KAM3.

If client C inadvertently sends a public key $X \equiv (p-1)$ to server S, a passive attacker $\mathcal{M}$ can perform off-line dictionary attacks on the communication messages $(X, Y, V_{C/S})$. When $X \equiv (p-1)$, a password-entangled public key Y (computed and sent by server S) and a shared secret key $Z_{C/S}$ (included in an authenticator $V_{C/S}$) will be as follows:

$$\begin{aligned} Y &\equiv ((p-1)^{u_1} \cdot V)^y \equiv (-1)^{u_1 \cdot y} \cdot (g^v)^y \\ &\equiv \left((-1)^{\frac{u_1 \cdot y}{v}} \cdot g^y\right)^v \end{aligned}$$

and

$$Z_{C/S} \equiv ((p-1) \cdot g^{u_2})^y \equiv (-1)^y \cdot g^{y \cdot u_2}$$

where $u_2 = \mathcal{H}(2\|X\|Y)$. Note that $X \equiv (p-1)$ is a valid value in the check of Server validation 1. Also, it is clear that $X \equiv (p-1)$ exists since the discrete logarithm of X is in the range $[1, q-1]$.

Because the server's private key y is randomly selected, the attacker $\mathcal{M}$ can test if $V_C \stackrel{?}{=} \overline{\mathcal{H}}(1\|X\|Y\| \pm Y^{u_2/v'})$ for all possible values (i.e., passwords) $v' = \mathcal{H}(pw')$. After these tests, attacker $\mathcal{M}$ finally finds out the correct client's password pw. Though the probability of $X \equiv (p-1)$ is negligible in the security parameter for $\mathbb{G}$, this attack might be meaningful because it is possible without knowing the discrete logarithm of X.

A countermeasure to the above attack is clear in the check of Server validation 1: If $X \notin [1, p-2]$, it outputs "invalid" and stops. Note that $X \equiv 1$ does not exist in the passive attack, and any active attacks are not possible when $X \equiv 1$.

5.3 In Summary

By combining the results of Section 5.1 and 5.2, we have a new sanity check for KAM3.

Step 1': Same as **Step 1** of Section 4.2

Step 2': After receiving (C, X), server S checks whether the client's public key X is in a specific range or not. If $X \notin [1, p-2]$, it outputs "invalid" and stops (<u>Server validation 1</u>). Otherwise, server S selects a random private key y from the range $[1, q-1]$ and computes a password-entangled public key $Y \equiv (X^{u_1} \cdot V)^y$ where $u_1 = \mathcal{H}(1\|X)$ and V is the client's password verification data. Then, server S sends the second message (S, Y) to client C.

$$S \rightarrow C : (S, Y)$$

Step 3': After receiving (S, Y), client C checks whether the server's public key Y is in a specific range or not. If $Y \notin [2, p-2]$, it outputs "invalid" and stops (Client validation 1). Otherwise, client C computes a shared secret key $Z_C \equiv Y^{(x+u_2)/(x \cdot u_1+v)}$, where $u_1 = \mathcal{H}(1\|X)$, $u_2 = \mathcal{H}(2\|X\|Y)$ and $v = \mathcal{H}(pw)$, and an authenticator $V_C = \overline{\mathcal{H}}(1\|X\|Y\|Z_C)$. Then, client C sends the third message V_C to server S.

$$C \rightarrow S : V_C$$

Step 4': Same as **Step 4** of Section 4.2

Step 5': Same as **Step 5** of Section 4.2

As said before, this sanity check is sufficient to prevent any possible attacks, discussed in Section 5.1 and 5.2.

参考文献

[1] S. M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attacks", In *Proc. of IEEE Symposium on Security and Privacy*, pp. 72-84, IEEE Computer Society, 1992.

[2] S. M. Bellovin and M. Merritt, "Augmented Encrypted Key Exchange: A Password-based Protocol Secure against Dictionary Attacks and Password File Compromise", In *Proc. of ACM CCS'93*, pp. 244-250, ACM Press, 1993.

[3] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated Key Exchange Secure against Dictionary Attacks", In *Proc. of EUROCRYPT 2000*, LNCS 1807, pp. 139-155, Springer-Verlag, 2000.

[4] C. Gentry, P. MacKenzie, and Z. Ramzan, "A Method for Making Password-Based Key Excahnge Resilient to Server Compromise", In *Proc. of CRYPTO 2006*, LNCS 4117, pp. 142-159, Springer-Verlag, 2006.

[5] Submissions to IEEE P1363.2. Available at `http://grouper.ieee.org/groups/1363/passwdPK/submissions.html`.

[6] IEEE 1363a, "IEEE Standard Specifications for Public-Key Cryptography−Amendment 1: Additional Techniques", IEEE Std 1363a$^{\text{TM}}$-2004, IEEE Computer Society, September 2004.

[7] IEEE 1363.2, "IEEE Standard Specifications for Password-Based Public-Key Cryptographic Techniques", IEEE Std 1363.2$^{\text{TM}}$-2008, IEEE Computer Society, January 2009.

[8] ISO/IEC 11770-4, "Information Technology−Security Techniques−Key Management−Part 4: Mechanisms Based on Weak Secrets", International Standard ISO/IEC 11770-4:2006(E), May 2006.

[9] D. Jablon, "Password Authentication Using Multiple Servers", In *Proc. of CT-RSA 2001*, LNCS 2020, pp. 344-360, Springer-Verlag, 2001.

[10] Research Papers on Password-based Cryptography. Available at `http://www.jablon.org/passwordlinks.html`.

[11] T. Kwon, "Ultimate Solution to Authentication via Memorable Password", IEEE P1363.2: Password-Based Public-Key Cryptography, May 2000.

[12] T. Kwon, "Authentication and Key Agreement via Memorable Password", In *Proc. of Network and Distributed System Security (NDSS) Symposium*, 2001.

[13] T. Kwon, "Summary of AMP (Authentication and Key Agreement via Memorable Passwords)", IEEE P1363.2: Password-Based Public-Key Cryptography, August 2003.

[14] T. Kwon, "Addendum to Summary of AMP", IEEE P1363.2: Password-Based Public-Key Cryptography, November 2003.

[15] T. Kwon, "Revision of AMP in IEEE P1363.2 and ISO/IEC 11770-4", IEEE P1363.2: Password-Based Public-Key Cryptography, June 2005.

[16] T. Wu, "The SRP Authentication and Key Exchange System", IETF RFC 2945, September 2000.