Security Analysis of J-PAKE

Mohsen Toorani
Department of Informatics
University of Bergen

***Abstract*—J-PAKE is a Password-Authenticated Key Exchange protocol, proposed in 2008 and presented again in 2010 and 2011. It does not require any public key infrastructure but uses zero-knowledge proofs. J-PAKE has been submitted as a candidate for the IEEE P1363.2 standard for password-based public key cryptography, and included in OpenSSL and OpenSSH. Since December 2010, J-PAKE has been used in Mozilla Firefox web browser. In this paper, we show that J-PAKE is vulnerable to replay attack, unknown key-share attack, and password compromise impersonation attack. We also propose some improvements for thwarting replay and unknown key-share attacks.**

I. Introduction

Password-Authenticated Key Exchange (PAKE) protocols enable two or more entities to authenticate each other and share a strong cryptographic key based on a pre-shared human memorable password. The first PAKE protocol, called Encrypted Key Exchange (EKE), was introduced by Bellowin and Meritt in 1992 [1]. Since introduction of the EKE, many PAKE protocols have been proposed. Many of those protocols have been shown to be insecure [2]–[5].

Based on number of participants, a PAKE protocol can be categorized into *two-party*, *three-party* or *four-party* setting. In the *two-party* setting, the participants are two entities, usually a client and a server that have shared a password. In the *three-party* setting (C2C-PAKE), there are two clients that have shared passwords with a trusted server, and the goal is to have a PAKE between clients. In the *four-party* setting (cross-realm C2C-PAKE), two clients have shared passwords with different servers, and they want to have a PAKE.

Password-Authentication Key Exchange by Juggling (J-PAKE) is a two-party PAKE protocol that was initially proposed in [6] and presented again in [7] and [8]. J-PAKE does not require any Public Key Infrastructure (PKI) but uses Zero-Knowledge Proofs (ZKP). Since 2008, J-PAKE has been available on website of the IEEE P1363.2 project for standard specifications of password-based public-key cryptography [9]. As it is mentioned in [10], no attack has been reported yet. The J-PAKE protocol has also been included in OpenSSL and OpenSSH, but a problem was reported on its implementations [11]. Since December 2010, J-PAKE has been used in Mozilla Firefox 4 web browser (beta 8 and later) [10].

In this paper, it is shown that the J-PAKE protocol [6]–[8] is vulnerable to a password compromise impersonation attack, and has other shortcomings regarding replay and Unknown Key-Share (UKS) attacks. Section II describes security attributes that should be provided by a PAKE protocol. Section III briefly reviews the J-PAKE protocol, and Section IV explains its security vulnerabilities. Section V comments on using J-PAKE in Mozilla Firefox web browser, and Section VI concludes the paper.

II. Security Requirements

There are some security attributes that PAKE protocols should possess [5], [12], [13]. It is desirable for PAKE protocols to provide the following security attributes:

- **Resilience to dictionary attack:** A PAKE protocol should not reveal any information that can be used for an offline/online dictionary attack. In an offline dictionary attack, the adversary eavesdrops communication between two honest entities and uses a dictionary of most probable passwords to obtain the password using the eavesdropped information. In an online dictionary attack, the adversary uses a dictionary of passwords but checks the validity of his or her guess through online communication with the target. For preventing online dictionary attacks, servers usually lock accounts of clients after several unsuccessful trials. However, there is a more complicated kind of this attack that is called an undetectable online dictionary attack, in which the adversary runs the protocol and

A variant of this article has been published in proceedings of the 19th IEEE Symposium on Computers and Communications (ISCC'14), Copyright ©IEEE. DOI: 10.1109/ISCC.2014.6912576

gets information that can be used for checking probable passwords offline. Then, the server cannot detect the attack.

- **Resilience to replay attack:** In a replay attack, an attacker that eavesdropped messages from previous runs of the protocol, replays them to impersonate an entity or gain another benefit.
- **Resilience to Unknown Key-Share attack:** Any PAKE protocol should be resilient to the UKS attack. That is, entity $\mathcal{A}$ should not be coerced into sharing a session key with $\mathcal{B}$ without $\mathcal{A}$'s knowledge so that $\mathcal{A}$ believes the key is shared with $\mathcal{M}$ and $\mathcal{B}$ correctly believes the key is shared with $\mathcal{A}$.
- **Resilience to password compromise impersonation attack:** When the password of entity $\mathcal{A}$ is disclosed, adversary $\mathcal{M}$ that has $\mathcal{A}$'s password can impersonate $\mathcal{A}$ and communicate with other entities, but it should not enable $\mathcal{M}$ to impersonate another honest entity and share a session key with $\mathcal{A}$. It is a stronger notion of security that is provided by some PAKE protocols.
- **Mutual Authentication:** The protocol should provide mutual authentication so that all the participants authenticate each other. Mutual authentication can thwart the man-in-the-middle attack.
- **Key control:** All the intended participants should be involved in calculation of the session key. No entity should be able to enforce the session key to fall into a pre-determined interval.
- **Known-key security:** Known-key security preserves the security of session keys after disclosure of a session key. Disclosure of a session key should not jeopardize the security of other session keys.
- **Resilience to Denning-Sacco attack:** It prevents an adversary to recover or guess the password (or long-term secret values) upon disclosure of a session key.
- **Forward secrecy:** Forward secrecy preserves the security of session keys after disclosure of the password. A PAKE protocol is forward secure if previous session keys remain secure even after disclosure of the password.

III. Review of the J-PAKE protocol

J-PAKE's specifications [6]–[8] provide an ambiguous description of the protocol. It presents a high-level description that requires zero-knowledge proofs, and suggests using the Schnorr's signature [14] for ZKPs. Figure 1 depicts the main description of the J-PAKE protocol. J-PAKE requires four passes of communication between two communicating entities, Alice and Bob, but the protocol can be completed in two rounds. There is not any provision for the implicit key confirmation so for having the key confirmation, two extra passes will be required. In the rest of this paper, Alice, Bob and the adversary will be denoted by $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{M}$, respectively.

Let G denotes a subgroup of $\mathbb{Z}_p^*$ of prime order q where p is prime and q is big enough for intractability of the Decisional Diffie-Hellman problem (DDH). Let g be a generator in G, and both $\mathcal{A}$ and $\mathcal{B}$ agree on (G, g). They also have a shared non-empty password $s \neq 0$ that its value falls within $[1, q-1]$. As mentioned in [8], s may also be a hash of the shared password together with some salt. Steps for high-level description of the J-PAKE protocol can be followed as:

1) $\mathcal{A}$ selects two random numbers x_1 and x_2 so that $x_1 \in_R [0, q-1]$ and $x_2 \in_R [1, q-1]$. She computes $X_1 = g^{x_1}$ and $X_2 = g^{x_2}$, and generates zero-knowledge proofs for x_1 and x_2. $\mathcal{A}$ sends X_1, X_2, and knowledge proofs for x_1 and x_2 to $\mathcal{B}$.
2) $\mathcal{B}$ selects two random numbers x_3 and x_4 so that $x_3 \in_R [0, q-1]$ and $x_4 \in_R [1, q-1]$. He computes $X_3 = g^{x_3}$ and $X_4 = g^{x_4}$, and generates zero-knowledge proofs for x_3 and x_4. $\mathcal{B}$ sends X_3, X_4, and knowledge proofs for x_3 and x_4 to $\mathcal{A}$.
3) $\mathcal{A}$ verifies the received knowledge proofs for x_3 and x_4. As x_4 should not be zero, $\mathcal{A}$ verifies if $X_4 \neq 1$. If verifications are confirmed, she computes $X_5 = (X_1 X_3 X_4)^{x_2 \cdot s}$ and zero-knowledge proofs for $x_2 \cdot s$, and sends them to $\mathcal{B}$. Otherwise, $\mathcal{A}$ halts the protocol.
4) $\mathcal{B}$ verifies the received knowledge proofs for x_1 and x_2. As x_2 should not be zero, $\mathcal{B}$ verifies if $X_2 \neq 1$. If verifications are confirmed, he computes $X_6 = (X_1 X_2 X_3)^{x_4 \cdot s}$ and zero-knowledge proofs for $x_4 \cdot s$, and sends them to $\mathcal{A}$. Otherwise, $\mathcal{B}$ halts the protocol.
5) $\mathcal{A}$ verifies the received knowledge proofs for $x_4 \cdot s$. If it is verified, she computes $K = (\frac{X_6}{X_4{}^{x_2 \cdot s}})^{x_2}$ and generates the session key k as $k = H(K)$ in which H is a hash function. Otherwise, $\mathcal{A}$ halts the protocol.
6) $\mathcal{B}$ verifies the received knowledge proofs for $x_4 \cdot s$. If it is verified, he computes $K = (\frac{X_5}{X_2{}^{x_4 \cdot s}})^{x_4}$ and generates the session key k as $k = H(K)$. Otherwise, $\mathcal{B}$ halts the protocol.

By completion of successful verifications, $\mathcal{A}$ and $\mathcal{B}$ authen-

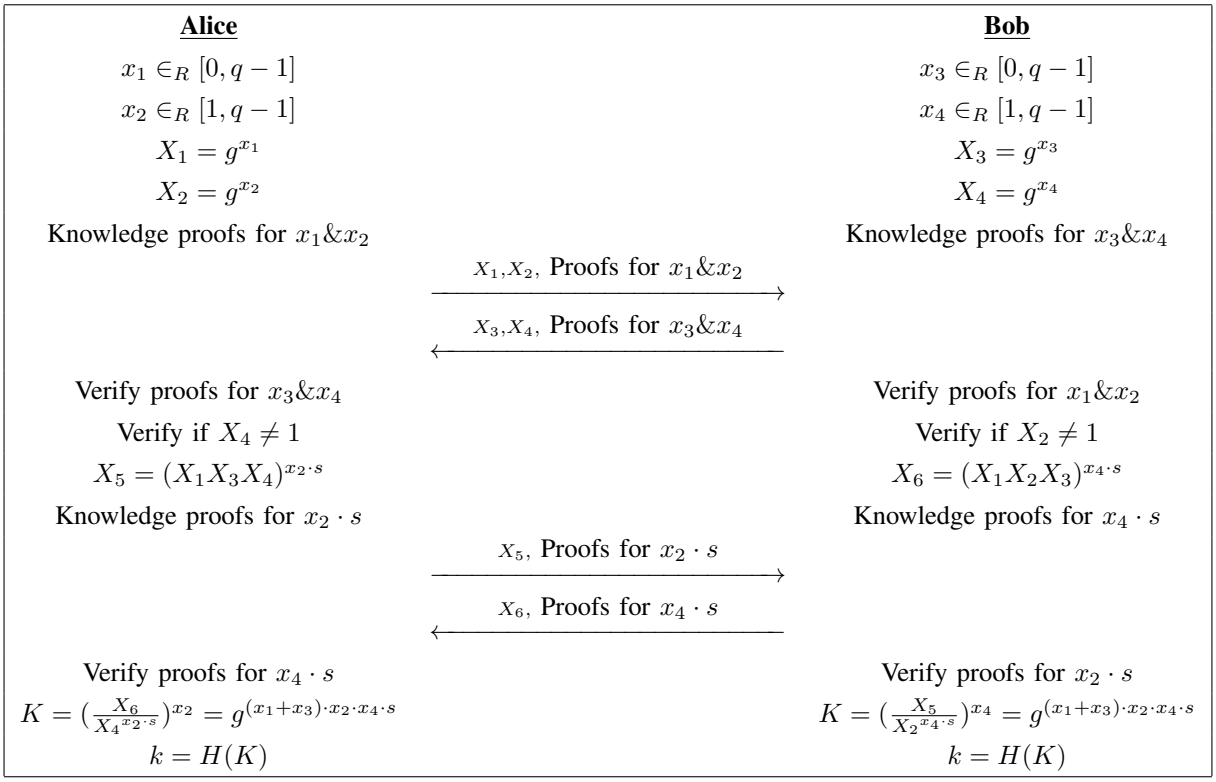

Fig. 1. The J-PAKE protocol as proposed in [6]–[8]

ticate each other and obtain the same session key k. The correctness can be simply verified as $K = (\frac{X_6}{X_4^{x_2 \cdot s}})^{x_2} = (\frac{X_5}{X_2^{x_4 \cdot s}})^{x_4} = g^{(x_1+x_3) \cdot x_2 \cdot x_4 \cdot s}$. Since the session key is computed as $k = H(K)$, and both $\mathcal{A}$ and $\mathcal{B}$ obtain the same value for K, they obtain the same session key. Steps (1), (3), and (5) are independent of steps (2), (4), and (6), respectively. Then, they can be done simultaneously at both sides, and the protocol can be completed in two rounds.

The main description of the J-PAKE protocol, depicted in Figure 1 includes fundamental blocks for zero-knowledge proofs and verifications without further details. J-PAKE's specifications [6]–[8] suggest using Schnorr's signature [14] for the zero-knowledge proofs. However, they do not provide a clear description of the protocol. Then, we provide a complete description of the J-PAKE protocol in Figure 2 in which ZKPs are substituted with the Schnorr's signature.

The Schnorr's signature scheme is provably secure in the random oracle model, which requires a secure hash function H. It has been used for variety of applications, from zero-knowledge proofs to signcryption schemes [15], [16]. To prove knowledge of x_1, that is the exponent in $X_1 = g^{x_1}$, $\mathcal{A}$ sends $\{ID_A, V_1 = g^{v_1}, r_1 = v_1 - x_1 h_1\}$ to $\mathcal{B}$ in which ID_A is the unique identifier of $\mathcal{A}$, $v_1 \in_R \mathbb{Z}_q$, and $h_1 = H(g, V_1, X_1, ID_A)$. Including ID_A in the calculation of h_1 is to prevent other entities from replaying $\mathcal{A}$'s signature back to $\mathcal{A}$. For verifying knowledge proofs for x_1, $\mathcal{B}$ verifies that X_1 lies in the prime-order subgroup G, and that $V_1 = g^{v_1}$ equals $g^{r_1}(X_1)^{h_1}$. Description of other knowledge proofs and verifications is somehow the same. The complete scheme is shown in Figure 2.

IV. Security Problems of J-PAKE

Resistance to offline and online dictionary attacks, forward secrecy, and known session key security are four general security requirements for key exchange protocols that are claimed for J-PAKE in [6]–[8]. However, there are no proofs of security, only heuristic ones. In [10], it is mentioned that no attack has been reported yet on the J-PAKE protocol. In this section, we show that J-PAKE lacks some desired security attributes of PAKE protocols that were explained in Section II.

A. Replay attack

J-PAKE has some features that can be used for a replay attack:

1) J-PAKE has four passes that can be concluded in two rounds. This means that two messages transmitted

Alice		**Bob**
$x_1 \in_R [0, q-1]$		$x_3 \in_R [0, q-1]$
$x_2 \in_R [1, q-1]$		$x_4 \in_R [1, q-1]$
$v_1 \in_R [0, q-1]$		$v_3 \in_R [0, q-1]$
$v_2 \in_R [0, q-1]$		$v_4 \in_R [0, q-1]$
$X_1 = g^{x_1}$		$X_3 = g^{x_3}$
$X_2 = g^{x_2}$		$X_4 = g^{x_4}$
$V_1 = g^{v_1}$		$V_3 = g^{v_3}$
$V_2 = g^{v_2}$		$V_4 = g^{v_4}$
$h_1 = H(g, V_1, X_1, ID_A)$		$h_3 = H(g, V_3, X_3, ID_B)$
$h_2 = H(g, V_2, X_2, ID_A)$		$h_4 = H(g, V_4, X_4, ID_B)$
$r_1 = v_1 - x_1 h_1$		$r_3 = v_3 - x_3 h_3$
$r_2 = v_2 - x_2 h_2$		$r_4 = v_4 - x_4 h_4$
	$\xrightarrow{ID_A, X_1, X_2, V_1, V_2, r_1, r_2}$	
	$\xleftarrow{ID_B, X_3, X_4, V_3, V_4, r_3, r_4}$	
Verify if $X_3, X_4 \in G$		Verify if $X_1, X_2 \in G$
Verify if $X_4 \neq 1$		Verify if $X_2 \neq 1$
$h_3 = H(g, V_3, X_3, ID_B)$		$h_1 = H(g, V_1, X_1, ID_A)$
$h_4 = H(g, V_4, X_4, ID_B)$		$h_2 = H(g, V_2, X_2, ID_A)$
Verify if $g^{r_3}(X_3)^{h_3} \stackrel{?}{=} V_3$		Verify if $g^{r_1}(X_1)^{h_1} \stackrel{?}{=} V_1$
Verify if $g^{r_4}(X_4)^{h_4} \stackrel{?}{=} V_4$		Verify if $g^{r_2}(X_2)^{h_2} \stackrel{?}{=} V_2$
$g_a = X_1 X_3 X_4$		$g_b = X_1 X_2 X_3$
$X_5 = {g_a}^{x_2 \cdot s}$		$X_6 = {g_b}^{x_4 \cdot s}$
$v_5 \in_R [0, q-1]$		$v_6 \in_R [0, q-1]$
$V_5 = {g_a}^{v_5}$		$V_6 = {g_b}^{v_6}$
$h_5 = H(g_a, V_5, X_5, ID_A)$		$h_6 = H(g_b, V_6, X_6, ID_B)$
$r_5 = v_5 - s x_2 h_5$		$r_6 = v_6 - s x_4 h_6$
	$\xrightarrow{ID_A, X_5, V_5, r_5}$	
	$\xleftarrow{ID_B, X_6, V_6, r_6}$	
Verify if $X_6 \in G$		Verify if $X_5 \in G$
$g_b = X_1 X_2 X_3$		$g_a = X_1 X_3 X_4$
$h_6 = H(g_b, V_6, X_6, ID_B)$		$h_5 = H(g_a, V_5, X_5, ID_A)$
Verify if ${g_b}^{r_6}(X_6)^{h_6} \stackrel{?}{=} V_6$		Verify if ${g_a}^{r_5}(X_5)^{h_5} \stackrel{?}{=} V_5$
$K = (\frac{X_6}{{X_4}^{x_2 \cdot s}})^{x_2}$		$K = (\frac{X_5}{{X_2}^{x_4 \cdot s}})^{x_4}$
$k = H(K)$		$k = H(K)$

Fig. 2. The J-PAKE protocol after substituting ZKP with Schnorr's signature

between $\mathcal{A}$ and $\mathcal{B}$ are completely independent of each other.

2) Messages exchanged between $\mathcal{A}$ and $\mathcal{B}$ can be divided into two distinct parts, one part for authentication and one part for the key exchange. Authentication credentials are independent of key exchange parts.
3) The key derivation function does not include any session specific information.

These features can be misused for a replay attack. This means that the adversary $\mathcal{M}$ can successfully thwart all

the authentications by replaying previous messages that are eavesdropped from previous runs of the protocol between $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{M}$ can impersonate $\mathcal{A}$ or $\mathcal{B}$ by replaying all the previous messages sent by $\mathcal{A}$ or $\mathcal{B}$, respectively. As the key confirmation is not included in J-PAKE's specifications and it is left optional, the authentication will be successful at the other party. The other party will then authenticate $\mathcal{M}$ as $\mathcal{A}$ or $\mathcal{B}$, respectively. Of course, the adversary cannot compute the session key, but all the authentications will be successful. It exhibits some weaknesses in the J-PAKE protocol. There can be some scenarios that the replay attack would work. For example, a naive applicant may suppose that the J-PAKE is a secure protocol, and use it just for the authentication purpose. Assuming that $\mathcal{M}$ impersonates $\mathcal{A}$, steps for a replay attack can be followed as:

1) $\mathcal{M}$ sends the eavesdropped message $\{ID_A, X_1', X_2', V_1', V_2', r_1', r_2'\}$ from a previous run of the protocol to $\mathcal{B}$.
2) $\mathcal{B}$ selects random numbers $\{x_3, x_4, v_3, v_4\}$, and computes $\{X_3, X_4, V_3, V_4, h_3, h_4, r_3, r_4\}$. $\mathcal{B}$ sends $\{ID_B, X_3, X_4, V_3, V_4, r_3, r_4\}$ to $\mathcal{M}$.
3) $\mathcal{M}$ sends the eavesdropped message $\{ID_A, X_5', V_5', r_5'\}$ from a previous run of the protocol to $\mathcal{B}$.
4) $\mathcal{B}$ verifies that $X_1', X_2' \in G$, and $X_2' \neq 1$. $\mathcal{B}$ computes $h_1' = H(g, V_1', X_1', ID_A)$ and $h_2' = H(g, V_2', X_2', ID_A)$, and verifies that $g^{r_1'}(X_1')^{h_1'} = V_1'$ and $g^{r_2'}(X_2')^{h_2'} = V_2'$. As the replayed messages are eavesdropped from a successful run of the protocol, both verifications will be successful. $\mathcal{B}$ generates random number v_6, computes $\{X_6 = (X_1'X_2'X_3)^{x_4 \cdot s}, V_6, h_6, r_6\}$, and sends $\{ID_B, X_6, V_6, r_6\}$ to $\mathcal{M}$.
   $\mathcal{B}$ also verifies that $X_5' \in G$ and computes h_5'. $\mathcal{B}$ verifies that ${g_a'}^{r_5'}(X_5')^{h_5'} = V_5'$. As the replayed message is eavesdropped from a successful run of protocol, the equation will be satisfied. $\mathcal{B}$ believes that $\mathcal{M}$ is $\mathcal{A}$. $\mathcal{B}$ computes $K = (\frac{X_5'}{X_2'^{x_4 \cdot s}})^{x_4}$, and generates the session key as $k = H(K)$.
5) $\mathcal{M}$ does not have s and x_2' so $\mathcal{M}$ cannot compute the session key but could successfully impersonate $\mathcal{A}$.

Similar steps can be written for the case that $\mathcal{M}$ impersonates $\mathcal{B}$. If J-PAKE is followed by further steps for key confirmation, which requires two extra passes, the replay attack would be detected. Another solution would be to modify h_5 and h_6.

Another scenario for the replay attack is through the so-called *key-replication* attack which invalidates the semantic security of the J-PAKE protocol in any security model that such an attack is allowed. In a key-replication attack [17], an attacker establishes two different non-matching sessions that output the same session key, in which one of the two sessions (the test session) is unexposed. If such an attack is successful, the adversary can easily find the session key of the test session by querying the second non-matching session which contains the same session key [17]. It is trivial to show that the J-PAKE protocol is vulnerable to the key-replication attack. The adversary $\mathcal{M}$ just needs to start two different sessions with $\mathcal{A}$ and $\mathcal{B}$ simultaneously by impersonating $\mathcal{B}$ and $\mathcal{A}$, respectively. The adversary just needs to forward messages that receives from one party to the other party. The established session key of these two sessions will be the same. As two established sessions are non-matching, $\mathcal{M}$ can query the second session which contains the same session key, and obtain the session key by issuing a $Reveal$ query. As the session keys of two non-matching sessions are the same, $\mathcal{M}$ could indeed obtain the session key of the test session. The J-PAKE protocol is then insecure in all the security models that a successful key-replication attack does not violate the freshness condition. A countermeasure to the key-replication attack is to include the unique session identifiers (SIDs) in the key derivation function [18].

B. Unknown Key-Share attack

The session key derivation function of the J-PAKE protocol does not include identifiers of $\mathcal{A}$ and $\mathcal{B}$. The session key is simply calculated as $k = H(K)$ in which $K = g^{(x_1+x_3)\cdot x_2 \cdot x_4 \cdot s}$. Then, there is not any binding between the session key and identifiers of participants. This can be used for an *Unknown Key-Share (UKS) attack*. The principle description of the J-PAKE protocol, presented in Figure 1 is potentially susceptible to the UKS attack, as there is no specific binding between identifiers of participants and authentication credentials, and the session key derivation function in the J-PAKE protocol. However, if we use the Schnorr's signature for ZKPs with specifications explained in Figure 2, identifiers of $\mathcal{A}$ and $\mathcal{B}$ are involved in computations of $\{h_1, h_2, h_5\}$ and $\{h_3, h_4, h_6\}$, respectively. Then, it is not vulnerable to the UKS attack. However, if one wants to use another ZKP instead of Schnorr's signature with specifications in Figure 2, it can make the protocol vulnerable to the UKS attack. A simple solution that can guarantee invulnerability of the J-PAKE protocol to the UKS

attack is to modify the session key derivation function as $k = H(K, ID_A, ID_B)$.

C. Password Compromise Impersonation attack

It is desirable for PAKE protocols to be resilient to Password Compromise Impersonation (PCI) attack. That is, even with compromise of $\mathcal{A}$'s low-entropy password s, $\mathcal{M}$ should not be able to impersonate $\mathcal{B}$ and share a session key with $\mathcal{A}$. Of course, disclosure of $\mathcal{A}$'s password allows an adversary to impersonate $\mathcal{A}$. However, this loss should not enable an adversary to impersonate other uncorrupted entities to $\mathcal{A}$.

Resilience to the PCI attack is a stronger notion of security that is provided by some PAKE protocols. Traditional two-party PAKE protocols were proved secure in the BPR2000 model [19] where the session key security is only based on the password security. Once the password is compromised, the security will be jeopardized. Results from the BPR2000 model is disputed in [5]. Regardless, the assumption for password security is not a realistic and practical assumption. In practice, people use the same password and sometimes they use a weakly selected password for several applications. Then, it is more practical to assume that passwords can be compromised by an adversary, and the PCI attack is not only of theoretical importance. For example, the adversary would impersonate a banking system and cause $\mathcal{A}$ to accept a session key, and then obtain her credit card number over the established session. This example demonstrates how PCI can lead to undesirable consequences. Therefore, it is very significant to design protocols that are secure against the PCI attack [20].

The first two-party PAKE protocol that could resist the PCI attack was introduced by Hitchcock et al. [21] that was proved to be secure in the CK2001 model [22]. However, a problem with its proofs was reported later in [23]. The J-PAKE does not have any security proofs. Then, its security requirements are not confined to any model. There are some PAKE protocols that provide resilience to the PCI attack. Then, for having a practical security comparison, it would be reasonable to show that the J-PAKE protocol is vulnerable to the PCI attack. In the following attack scenario, we assume that an adversary $\mathcal{M}$ could obtain the password s of client $\mathcal{A}$. The goal of adversary is then to share a session key with $\mathcal{A}$ by masquerading as $\mathcal{B}$. Here is the attack scenario:

- $\mathcal{A}$ or $\mathcal{M}$ start the protocol. $\mathcal{A}$ selects x_1 and x_2, computes $\{X_1,\ X_2,\ \textit{ZKP for}\ x_1 \& x_2\}$, and sends them to $\mathcal{M}$. $\mathcal{M}$ selects x_3 and x_4. $\mathcal{M}$ computes $\{X_3,\ X_4,\ \textit{ZKP for}\ x_3 \& x_4\}$, and sends them to $\mathcal{A}$.
- $\mathcal{A}$ verifies proofs for x_3 and x_4, and checks if $X_4 \neq 1$. $\mathcal{A}$ computes $\{X_5$ and ZKP for $x_2 \cdot s\}$, and sends them to $\mathcal{M}$. $\mathcal{M}$ computes $\{X_6$ and ZKP for $x_4 \cdot s\}$, and sends them to $\mathcal{A}$. As $\mathcal{M}$ is supposed to know s, he can generate ZKP for $x_4 \cdot s$.
- $\mathcal{A}$ verifies proofs for $x_4 \cdot s$. The authentication will be successful, and $\mathcal{A}$ computes K and k. $\mathcal{M}$ computes K and k but he does not need to authenticate $\mathcal{A}$. $\mathcal{M}$ could successfully impersonate $\mathcal{B}$ and share a session key with $\mathcal{A}$.

D. Further Defects

1) The J-PAKE protocol requires many computations that includes at least 28 exponentiations and 10 random number generations if the Schnorr's signature is used for zero-knowledge proofs. However, it was claimed for its efficiency in [6]–[8] where J-PAKE was partially compared with EKE [1] and SPEKE [3] protocols. EKE and SPEKE are the first, but not the most efficient PAKE protocols. Security of EKE and SPEKE is disputed in [2] and [4], respectively.
2) The first round of J-PAKE does not use any pre-shared secret, e.g. password. Then, anyone can successfully complete the first round. The server cannot detect a legitimate user from an attacker unless to the end of the protocol, just before calculating the session key. Form a practical security viewpoint, this can increase server's vulnerability to a *Denial of Service (DoS)* attack. The problem concerning authentication protocols and DoS attacks is well understood, and much work is invested to address it [24]. From the DoS attack perspective, it would be better to have knowledge proofs of password at the first round. Then, an attacker that did not share any password with the server, would be detected at the beginning.
3) As it is mentioned in [6]–[8], [10], J-PAKE is a *balanced* protocol, and is vulnerable to the *server compromise*. By *balanced* PAKE protocols, they meant protocols that allow participants to use the same password. By *augmented* PAKE protocols, they meant protocols that the server does not save clients' passwords in clear, and provides more security to the server compromise. The vulnerability of J-PAKE to server compromise is justified

by challenging the need for *augmented* PAKE protocols [6]–[8]: (1) None of previously proposed *augmented* PAKE protocols is really resilient to offline attacks when the server is compromised, and the password file on the server is stolen. (2) Even *balanced* PAKE protocols will be modified to avoid storing passwords in clear on servers when they are implemented, so they do not store plain passwords on servers. However, the argument is disputable as: (1) Many *augmented* PAKE protocols are based on storing hash of password together with information that vary for each client. After a server compromise, the attacker should perform a separate offline attack for each client. This would be a brute-force attack if the passphrase is random. The situation is different for *balanced* protocols where all clients' passwords are compromised immediately after the server compromise. (2) Although general methods like the Ω-*method* [25] are proposed to make PAKE protocols resilient to the *server compromise*, it does not stop working on augmented protocols. The Ω-method requires an extra round of communication, several hash calculations and a signature calculation/verification. Regardless, one would expect a protocol to express all the stages clearly. Modifications without scrutiny can make a protocol insecure.

4) As we have $r_5 = v_5 - sx_2h_5$ and $r_6 = v_6 - sx_4h_6$ in which r_5, h_5, r_6 and h_6 are publicly known, the practical security of J-PAKE depends on randomness and confidentiality of x_2 and v_5, or x_4 and v_6. If the attacker can affect random number generation at the user end or have access to the output of random number generator by some means, the password s can be easily calculated.

V. J-PAKE in Mozilla Firefox

Mozilla Firefox 4 beta 8 (released in December 2010) and its later versions use J-PAKE protocol for the Firefox Sync [10]. Firefox Sync is a service that lets users to synchronize bookmarks, cached passwords, preferences, open tabs, and history of visited sites on Firefox browsers on different devices. After setting up Firefox Sync on the host device, an account with 25 MB storage limitation is created for each user. A random *Sync-key* or *Recovery Key* of 128 bits is generated for each user which is represented as 26 characters in base32 alphabet. The Sync-key is then used for deriving the *Sync Key bundle* which consists of a 256-bit symmetric encryption key and a 256-bit HMAC key. The encryption key, along with a randomly generated 16-byte IV, is used in the AES algorithm to produce the ciphertext from the user's data that should be synchronized. The ciphertext and its HMAC are stored on Mozilla's servers [26]. User's browsers on different devices are then synchronized with Mozilla's servers. For doing this, all devices should be added to the user account on Mozilla's servers. For pairing a new device, the user should get a 12-character secret code from the host browser and insert it at the same time to the second browser. Using the secret code and the J-PAKE protocol, the sync-key is encrypted with a new key and transferred to the new device. An alternative approach, when the user is not near the first computer, is to use the username and password of sync account on Mozilla servers and the Recovery Key.

Although use of a PAKE protocol in this setting and for synchronization would be strange, there are some issues that should be considered. Maybe Mozilla Firefox has used J-PAKE because it is a patent-free protocol, but J-PAKE is a protocol without proofs of security. J-PAKE does not require PKI but it does not meet those notions of security that can be attained by those PAKE protocols that would require the server to have a public key [27]. It is noteworthy that any web server that supports TLS/SSL protocol, already has a public key, uses X.509 certificates, and is then involved in a PKI. As the synchronization is based on working in the cloud, selecting an efficient and secure PAKE protocol that would require the server to have a public key, should not cause any serious problem or limitation.

VI. Conclusion

Security of the J-PAKE protocol [6]–[8] was analyzed in this paper. We showed that J-PAKE is vulnerable to replay, UKS, and password compromise impersonation attacks. Some improvements have also been proposed for thwarting replay and UKS attacks. J-PAKE requires at least 28 exponentiations and 10 random number generations that makes it computationally costly. As password-based authentication is postponed to end of the J-PAKE protocol, it can increase server's vulnerability to the DoS attack. Application of J-PAKE in Mozilla Firefox Sync mechanism was also reviewed in this paper. Although J-PAKE provides kind of simplicity by avoiding certified public keys and PKI, it does not meet those notions of security that can be attained by those PAKE protocols that would require a server to have a public key.

REFERENCES

[1] S. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks," in *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, pp. 72–84, 1992.

[2] S. Patel, "Number theoretic attacks on secure password schemes," in *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pp. 236–247, May 1997.

[3] D. Jablon, "Strong Password-only Authenticated Key Exchange," *ACM SIGCOMM Computer Communication Review*, vol. 26, no. 5, pp. 5–26, 1996.

[4] M. Zhang, "Analysis of the SPEKE password-authenticated key exchange protocol," *IEEE Communications Letters*, vol. 8, no. 1, pp. 63–65, 2004.

[5] Z. Zhao, Z. Dong, and Y. Wang, "Security analysis of a password-based authentication protocol proposed to IEEE 1363," *Theoretical Computer Science*, vol. 352, no. 1, pp. 280–287, 2006.

[6] F. Hao and P. Ryan, "Password Authenticated Key Exchange by Juggling," *16th Workshop on Security Protocols*, 2008.

[7] F. Hao and P. Ryan, "Password Authenticated Key Exchange by Juggling," in *Proceedings of the 16th Security Protocols Workshop, LNCS 6615*, pp. 159–171, Springer, 2011.

[8] F. Hao and P. Ryan, "J-PAKE: Authenticated Key Exchange without PKI," *Transactions on Computational Science XI*, pp. 192–206, 2010.

[9] IEEE P1363.2. http://grouper.ieee.org/groups/1363/passwdPK/submissions.html.

[10] F. Hao and P. Ryan, "How To Sync with Alice," in *Proceedings of the 19th Security Protocols Workshop, LNCS 7114*, pp. 170–178, Springer, 2011.

[11] S. Martini, "Session Key Retrieval in J-PAKE Implementations of OpenSSL and OpenSSH." http://seb.dbzteam.org/crypto/jpake-session-key-retrieval.pdf, 2010.

[12] M. Toorani, "Cryptanalysis of a new protocol of wide use for email with perfect forward secrecy," *Security and Communication Networks*, 2014.

[13] J. Yang and T. Cao, "Provably Secure Three-party Password Authenticated Key Exchange Protocol in the Standard Model," *Journal of Systems and Software*, vol. 85, no. 2, pp. 340–350, 2012.

[14] C. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[15] M. Toorani and A. Beheshti, "An elliptic curve-based signcryption scheme with forward secrecy," *Journal of Applied Sciences*, vol. 9, no. 6, pp. 1025–1035, 2009.

[16] M. Toorani and A. Beheshti, "A directly public verifiable signcryption scheme based on elliptic curves," in *Proceedings of the 14th IEEE Symposium on Computers and Communications (ISCC'09)*, pp. 713–716, 2009.

[17] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol," in *Advances in Cryptology–CRYPTO'05*, pp. 546–566, Springer, 2005.

[18] K.-K. Choo, C. Boyd, and Y. Hitchcock, "On session key construction in provably-secure key establishment protocols," in *Progress in Cryptology – Mycrypt 2005*, vol. 3715 of *Lecture Notes in Computer Science*, pp. 116–131, Springer Berlin Heidelberg, 2005.

[19] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Advances in Cryptology – EUROCRYPT'00, LNCS 1807*, pp. 139–155, Springer, 2000.

[20] D. Feng and J. Xu, "A new client-to-client password-authenticated key agreement protocol," in *International Workshop on Coding and Cryptology (IWCC'09), LNCS 5557*, pp. 63–76, Springer, 2009.

[21] Y. Hitchcock, Y. S. T. Tin, J. M. Gonzalez-Nieto, C. Boyd, and P. Montague, "A password-based authenticator: Security proof and applications," in *Progress in Cryptology – INDOCRYPT'03, LNCS 2904*, pp. 388–401, Springer, 2003.

[22] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Advances in Cryptology – EUROCRYPT'01, LNCS 2045*, pp. 453–474, Springer, 2001.

[23] K.-K. R. Choo, C. Boyd, and Y. Hitchcock, "Errors in computational complexity proofs for protocols," in *Advances in Cryptology – ASIACRYPT'05, LNCS 3788*, pp. 624–643, Springer, 2005.

[24] D. Park, J. Kim, C. Boyd, and E. Dawson, "Cryptographic salt: A countermeasure against denial-of-service attacks," in *Proceedings of 6th Australasian Conference on Information Security and Privacy (ACISP'01), LNCS 2119*.

[25] C. Gentry, P. MacKenzie, and Z. Ramzan, "A method for making password-based key exchange resilient to server compromise," in *Advances in Cryptology – CRYPTO 2006, LNCS 4117*, pp. 142–159, Springer, 2006.

[26] Mozilla Services. http://docs.services.mozilla.com/sync/overview.html, Accessed 2013.

[27] S. Halevi and H. Krawczyk, "Public-key cryptography and password protocols," *ACM Transactions on Information and System Security (TISSEC)*, vol. 2, no. 3, pp. 230–268, 1999.