SECURITY ANALYSIS OF J-PAKE

Mohsen Toorani

J-PAKE is a password-authenticated key exchange protocol in the two-party setting where participants have only a shared password without any public key. The J-PAKE protocol has been submitted as a candidate to the IEEE P1363.2 standard for password-based public key cryptography, and included in OpenSSL and OpenSSH. Since December 2010, it has been used in Mozilla Firefox web browser. In this paper, we show that the plain J-PAKE protocol is vulnerable to unknown key-share and key-replication attacks, and has other shortcomings with respect to other PAKE protocols. We also propose some improvements.

1 Introduction

Password-Authenticated Key Exchange (PAKE) protocols enable two or more entities to authenticate each other, and share a 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 have security vulnerabilities [2–6], even though some of them were accompanied by provable security. The BPR model [7] was the first game-based security model that was specifically introduced for password-based protocols. However, the BPR model in its original formulation is shown to be a weak security model that might not capture some attacks [2].

Based on number of participants, a PAKE protocol can be categorized into two-party, three-party, four-party, or multi-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 typically two clients that have shared passwords with a trusted server, and the goal is to have an authenticated key establishment between clients. In the *four-party* setting (cross-realm C2C-PAKE), two clients have shared passwords with different servers, and they want to have an authenticated key establishment.

Password-Authenticated Key Exchange by Juggling (J-PAKE) [8, 9] is a two-party PAKE protocol with password-only authentication. It 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 [10]. 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) [12].

In this paper, we perform a security analysis on the J-PAKE protocol, and show that it has some shortcomings. Section 2 describes security attributes that should be provided by a PAKE protocol. Section 3 briefly reviews the J-PAKE protocol, and Section 4 explains its security vulnerabilities. Section 5 comments on using J-PAKE in Mozilla Firefox web browser, and Section 6 concludes the paper.

2 Security Requirements

There are some security attributes that PAKE protocols should possess [6, 13, 14]. 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 guess through online communication with the target. For preventing online dictionary attacks, servers usually lock accounts of clients after several unsuccessful trials.

- **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 $\mathcal{B}$ so that $\mathcal{A}$ believes that the key is shared with another entity $\mathcal{C} \neq \mathcal{B}$, while $\mathcal{B}$ believes that the key is shared with $\mathcal{A}$.
- **Mutual Authentication:** The protocol should provide mutual authentication so that all the participants authenticate each other.
- **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.

There are other desirable security attributes that are required for some applications. They can be considered as add-on in the corresponding security models for PAKE protocols. Such security guarantees may imply further requirements such as having public/priavte key at the server side [15] or may increase computational costs. Such extra security guarantees include resilience to password compromise impersonation attack [16], resilience to server compromise [17], and resilience to denial-of-service (DoS) attack [18].

3 Review of the J-PAKE protocol

J-PAKE's specifications [8, 9] provide an ambiguous description of the protocol. It presents a high-level description that requires some zero-knowledge proofs, and suggests using the Schnorr signature [19] 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, and the protocol can be completed in two rounds. There is not any provision for the implicit key confirmation. Then, for having the key confirmation, extra passes would 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 password $s \neq 0$ that its value falls within $[1, q-1]$. Without giving any more details, it is mentioned in [9] that s may also be a hash value 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 $x_2 \in_R [1, q-1]$. $\mathcal{A}$ 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 $\mathcal{B}$.

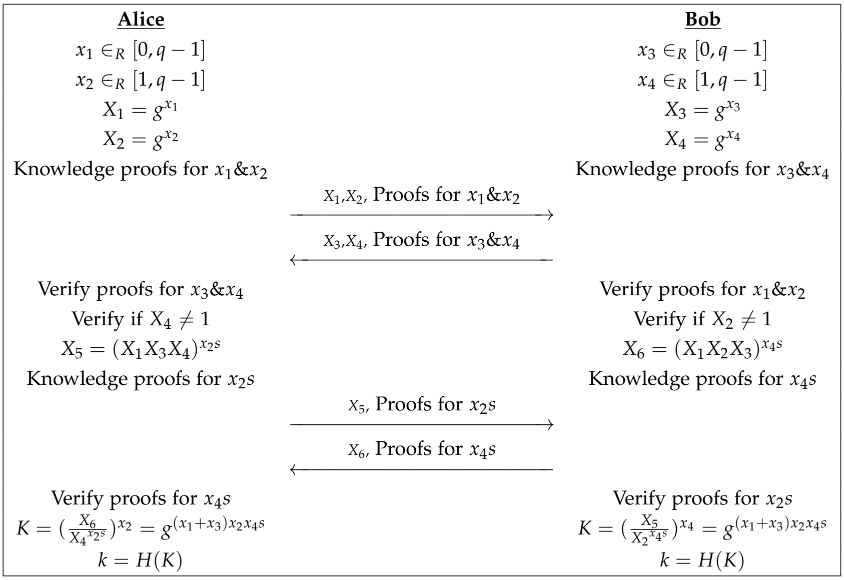
Alice
$x_1 \in_R [0, q-1]$
$x_2 \in_R [1, q-1]$
$X_1 = g^{x_1}$
$X_2 = g^{x_2}$
Knowledge proofs for $x_1 \& x_2$
Bob
$x_3 \in_R [0, q-1]$
$x_4 \in_R [1, q-1]$
$X_3 = g^{x_3}$
$X_4 = g^{x_4}$
Knowledge proofs for $x_3 \& x_4$
X_1, X_2, Proofs for $x_1 \& x_2$
X_3, X_4, Proofs for $x_3 \& x_4$
Verify proofs for $x_3 \& x_4$
Verify if $X_4 \neq 1$
$X_5 = (X_1 X_3 X_4)^{x_2 s}$
Knowledge proofs for $x_2 s$
Verify proofs for $x_1 \& x_2$
Verify if $X_2 \neq 1$
$X_6 = (X_1 X_2 X_3)^{x_4 s}$
Knowledge proofs for $x_4 s$
X_5, Proofs for $x_2 s$
X_6, Proofs for $x_4 s$
Verify proofs for $x_4 s$
$K = (\frac{X_6}{X_4^{x_2 s}})^{x_2} = g^{(x_1+x_3)x_2 x_4 s}$
$k = H(K)$
Verify proofs for $x_2 s$
$K = (\frac{X_5}{X_2^{x_4 s}})^{x_4} = g^{(x_1+x_3)x_2 x_4 s}$
$k = H(K)$

Fig. 1: *The J-PAKE protocol as proposed in [8, 9]*

$\mathcal{B}$ selects two random numbers x_3 and x_4 so that $x_3 \in_R [0, q-1]$ and $x_4 \in_R [1, q-1]$. $\mathcal{B}$ 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 $\mathcal{A}$.

2. $\mathcal{A}$ 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, $\mathcal{A}$ computes $X_5 = (X_1 X_3 X_4)^{x_2 s}$ and zero-knowledge proofs for $x_2 s$, and sends them to $\mathcal{B}$. Otherwise, $\mathcal{A}$ halts the protocol.

   $\mathcal{B}$ 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, $\mathcal{B}$ computes $X_6 = (X_1 X_2 X_3)^{x_4 s}$ and zero-knowledge proofs for $x_4 s$, and sends them to $\mathcal{A}$. Otherwise, $\mathcal{B}$ halts the protocol.

3. $\mathcal{A}$ verifies the received knowledge proofs for $x_4 s$. If it is verified, $\mathcal{A}$ computes $K = (\frac{X_6}{X_4^{x_2 s}})^{x_2}$, and generates the session key k as

$k = H(K)$ in which H is a hash function. Otherwise, $\mathcal{A}$ halts the protocol.

$\mathcal{B}$ verifies the received knowledge proofs for x_4s. If it is verified, $\mathcal{B}$ computes $K = (\frac{X_5}{X_2{}^{x_4s}})^{x_4}$, and generates the session key k as $k = H(K)$. Otherwise, $\mathcal{B}$ halts the protocol.

By completion of successful verifications, $\mathcal{A}$ and $\mathcal{B}$ authenticate each other and agree on the same session key k. The correctness can be simply verified as $K = (\frac{X_6}{X_4{}^{x_2s}})^{x_2} = (\frac{X_5}{X_2{}^{x_4s}})^{x_4} = g^{(x_1+x_3)x_2x_4s}$. 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. The J-PAKE protocol does not imply any ordering for $\mathcal{A}$ or $\mathcal{B}$ as the initiator of the protocol. $\mathcal{A}$ and $\mathcal{B}$ can accomplish each round simultaneously as neither party depend on the other [8, 9]. This means that either $\mathcal{A}$ or $\mathcal{B}$ can be the initiator of the protocol.

The Schnorr signature scheme includes a secure hash function H, and is provably secure in the random oracle model. It has been used for a variety of applications in the literature [20, 21]. The Schnorr identification scheme is zero-knowledge for an honest verifier, where an honest verifier is the one who selects the challenges at random. However, it is not zero-knowledge for a malicious verifier if the challenge is large. The scheme might be made non-interactive through a Fiat-Shamir transformation [22], assuming that there exists a secure cryptographic hash function. To prove the knowledge of the exponent x in $X = g^x$, an entity may send $\{SignerID, OtherInfo, U = g^u, r = u - xh\}$ in which $SignerID$ is the unique identifier of the entity, $OtherInfo$ may include auxiliary information, $u \in_R \mathbb{Z}_q$, and h maybe defined as $h = H(g, U, X, SignerID, OtherInfo)$ [8, 9]. For verifying knowledge proofs for x, the verifier computes h and verifies that $U = g^r X^h$. It is noteworthy that having $SignerID$ in calculation of h is not part of the original Schnorr scheme [19]. There is also an extra assumption in [8, 9] that a verifier in the Schnorr scheme will check that X lies in the subgroup of prime order q, although such verification is not part of the Schnorr scheme [19].

4 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 the J-PAKE protocol in [8, 9] through heuristic reasoning. In this section, we show that the J-PAKE protocol lacks some desired security attributes. We also propose some improvements.

4.1 Unknown Key-Share attack

The UKS attack is an attack against the authentication goal, whereby two entities will have different beliefs about their peers after a key establishment. The UKS attack is typically feasible when a key exchange protocol fails to provide an authenticated binding between the session key and identifiers of the involved entities [23]. Although the UKS attack is very important and practical, some security models such as the BPR model [7] may fail to capture an UKS attack [2]. For modeling an UKS attack on two-party password-only based PAKE protocols, it is reasonable to assume that a client has shared the same password with two servers [24]. Such assumption is realistic and according to the common practice where people select the same password on different servers. For two-party PAKE protocols that require the server to have a public key, the UKS attack might be conducted through other scenarios. For three-party PAKE protocols, there are more sophisticated ways to mount an UKS attack [2].

The main description of the J-PAKE protocol, depicted 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. The session key is simply calculated as $k = H(g^{(x_1+x_3)x_2x_4s})$ which does not include identifiers of the participants. This can lead to an UKS attack if *unique* identifiers are not included in ZKPs. A scenario for the UKS attack is depicted in Figure 2 where a client (C) has shared the same password s with two servers S_1 and S_2. As mentioned earlier, such assumption is practical. In Figure 2, it is also assumed that *unique* identifiers are not included in ZKPs. J-PAKE specification suggests using the Schnorr signature for ZKPs, but does not implement them in the protocol. An instance of the J-PAKE protocol may use another scheme for ZKPs without any binding to the identifiers, or may not use unique identifiers for participants. In some implementations of the J-PAKE protocol for the client-server scenario, $SignerID$ has been taken as "client" or "1" for all clients, and as "server" or "2" for all servers [25]. Then, different clients and servers will have the same identifiers in different sessions which makes the UKS attack feasible.

For completeness of an UKS attack, the adversary is not required to find the session key. In Figure 2, $\mathcal{M}$ does not find the session key. However, $\mathcal{M}$ can find the session key if the UKS attack is formalized through a game-based security experiment. It is straightforward to verify that two sessions that are separated by a dotted line in Figure 2, are non-matching because owners and peers of the sessions are different. As two sessions are non-matching and completed with the same session key, $\mathcal{M}$ can take one of those sessions as the test session, and ask a *Reveal* query for the other non-matching session. The *Reveal* query will then return the same session key as that of the test session.

Using unique identifiers of participants through non-interactive zero-knowledge (NIZK) proofs might thwart the UKS attack. However, a better solution that can guarantee invulnerability of the J-PAKE protocol to the UKS attack is to modify the session key derivation function as $k = H(K, \hat{\mathcal{A}}, \hat{\mathcal{B}})$ in which $\hat{\mathcal{A}}$, $\hat{\mathcal{B}}$ denotes unique identifiers of $\mathcal{A}$ and $\mathcal{B}$, respectively.

4.2 Key-replication attack

The *key-replication* attack [26, 27] can invalidate the semantic security of a protocol in any security model that such an attack is feasible. In a key-replication attack, an attacker establishes two different non-matching sessions so that they output the same session key. One of these non-matching sessions will be taken as the test session, which will remain unexposed. The other non-matching session will be queried by the adversary. The adversary issues a *Reveal* query for the second non-matching session, and obtains its session key. As the queried session has

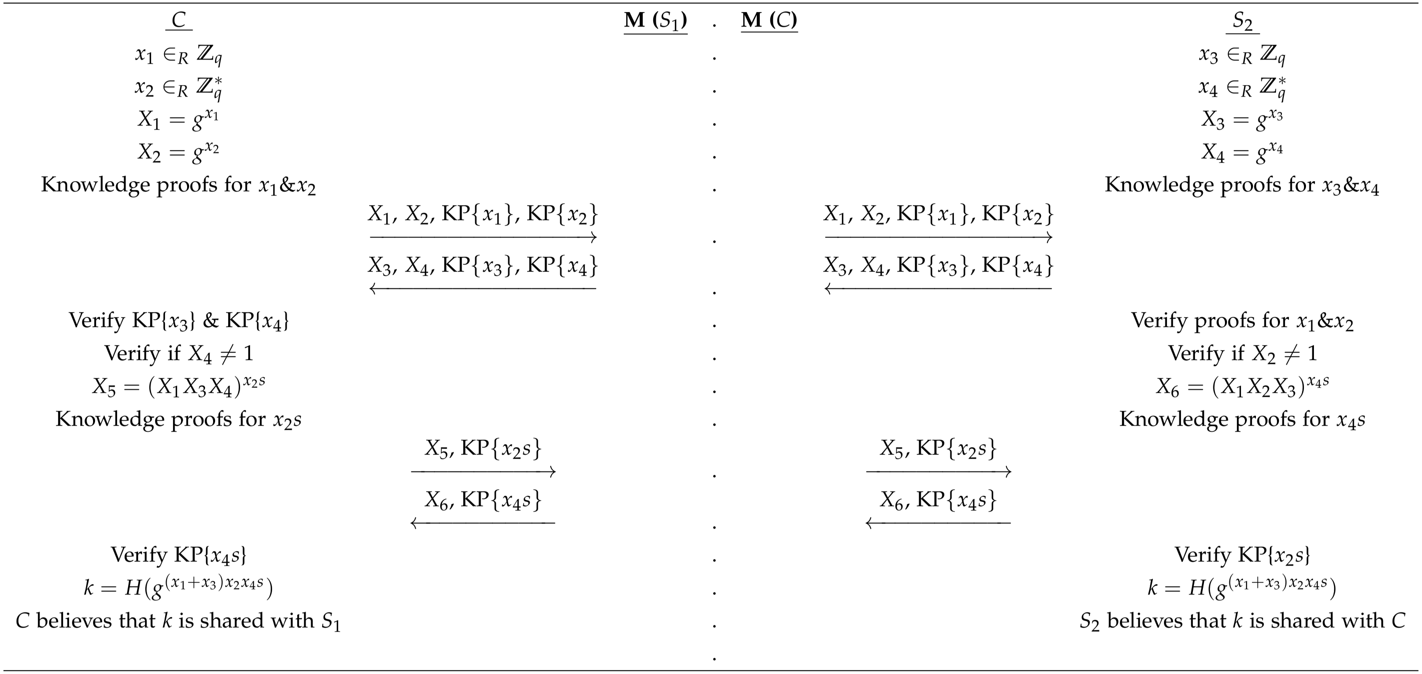
C
$x_1 \in_R \mathbb{Z}_q$
$x_2 \in_R \mathbb{Z}_q^*$
$X_1 = g^{x_1}$
$X_2 = g^{x_2}$
Knowledge proofs for $x_1 \& x_2$
X_1, X_2, KP$\{x_1\}$, KP$\{x_2\}$
X_3, X_4, KP$\{x_3\}$, KP$\{x_4\}$
Verify KP$\{x_3\}$ & KP$\{x_4\}$
Verify if $X_4 \neq 1$
$X_5 = (X_1 X_3 X_4)^{x_2 s}$
Knowledge proofs for $x_2 s$
X_5, KP$\{x_2 s\}$
X_6, KP$\{x_4 s\}$
Verify KP$\{x_4 s\}$
$k = H(g^{(x_1+x_3)x_2 x_4 s})$
C believes that k is shared with S_1
M (S_1)
M (C)
X_1, X_2, KP$\{x_1\}$, KP$\{x_2\}$
X_3, X_4, KP$\{x_3\}$, KP$\{x_4\}$
X_5, KP$\{x_2 s\}$
X_6, KP$\{x_4 s\}$
S_2
$x_3 \in_R \mathbb{Z}_q$
$x_4 \in_R \mathbb{Z}_q^*$
$X_3 = g^{x_3}$
$X_4 = g^{x_4}$
Knowledge proofs for $x_3 \& x_4$
Verify proofs for $x_1 \& x_2$
Verify if $X_2 \neq 1$
$X_6 = (X_1 X_2 X_3)^{x_4 s}$
Knowledge proofs for $x_4 s$
Verify KP$\{x_2 s\}$
$k = H(g^{(x_1+x_3)x_2 x_4 s})$
S_2 believes that k is shared with C

Fig. 2: *A practical scenario for the UKS attack on the J-PAKE protocol where C is assumed to have the same password on two different servers S_1 and S_2. Finally, C incorrectly believes that k is shared with S_1, while S_2 correctly believes that k is shared with C.*

the same session key as of the test session, the adversary could indeed obtain the session key of the test session which is still unexposed. This invalidates the semantic security of the protocol in any security model that such an attack does not violate the freshness condition of the test session.

As mentioned in Section 3, in the J-PAKE protocol, either $\mathcal{A}$ or $\mathcal{B}$ can be the initiator. It is trivial to show that the J-PAKE protocol is vulnerable to the key-replication attack if we allow the protocol to run in the lockstep. A scenario for the key-replication attack is depicted in Figure 3. By issuing two *Send* queries, the adversary $\mathcal{M}$ orders $\mathcal{A}$ and $\mathcal{B}$ to initiate two different sessions. Let oracle $\prod_A^i$ denote instance i of principle $\mathcal{A}$. Let oracle $\prod_B^j$ denote instance j of principle $\mathcal{B}$. $\mathcal{M}$ then asks consequent *Send* queries to $\prod_A^i$ and $\prod_B^j$ wherein response of *Send* query from one oracle is placed into the *Send* query to the other oracle. This looks like replaying messages between two parallel sessions, as it is depicted in Figure 3. Finally, the established session keys of these two sessions will be the same. As two established sessions are non-matching, $\mathcal{M}$ can query the second non-matching session which contains the same session key, and obtain the session key of the test session 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 which violates the semantic security of the J-PAKE protocol in any security model wherein two sessions depicted in Figure 3 are non-matching and do not violate the freshness condition of that security model. This includes the BPR model [7] if we assume the protocol to run in the lockstep and define the session identifiers as transcripts of the protocol run as they appear in the protocol. A simple countermeasure to the key-replication attack is to modify the key derivation function so that it includes the session identifiers [27]. The session identifiers can be defined as concatenation of messages exchanged during the protocol run. Then, two non-matching sessions will not yield the same session key, and it will thwart the key-replication attack.

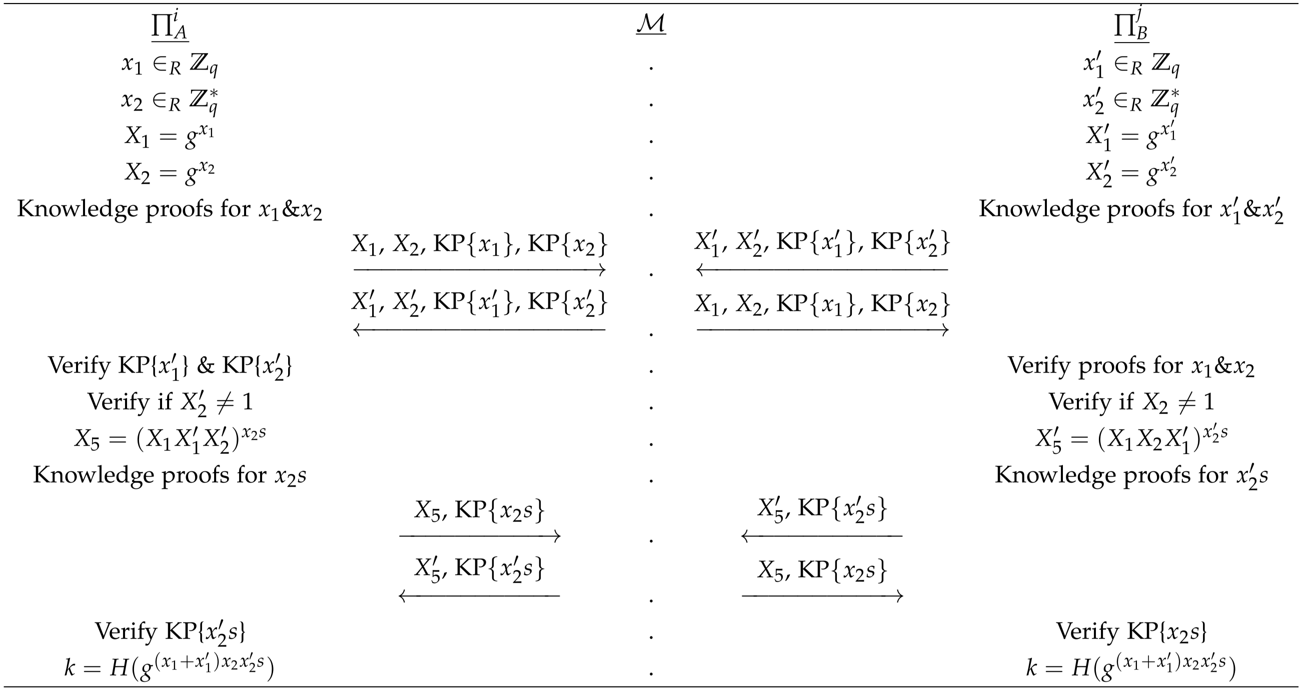
Π_A^i
$\mathcal{M}$
Π_B^j
$x_1 \in_R \mathbb{Z}_q$
$x'_1 \in_R \mathbb{Z}_q$
$x_2 \in_R \mathbb{Z}_q^*$
$x'_2 \in_R \mathbb{Z}_q^*$
$X_1 = g^{x_1}$
$X'_1 = g^{x'_1}$
$X_2 = g^{x_2}$
$X'_2 = g^{x'_2}$
Knowledge proofs for $x_1 \& x_2$
Knowledge proofs for $x'_1 \& x'_2$
$X_1, X_2, \mathrm{KP}\{x_1\}, \mathrm{KP}\{x_2\}$
$X'_1, X'_2, \mathrm{KP}\{x'_1\}, \mathrm{KP}\{x'_2\}$
$X'_1, X'_2, \mathrm{KP}\{x'_1\}, \mathrm{KP}\{x'_2\}$
$X_1, X_2, \mathrm{KP}\{x_1\}, \mathrm{KP}\{x_2\}$
Verify $\mathrm{KP}\{x'_1\}$ & $\mathrm{KP}\{x'_2\}$
Verify proofs for $x_1 \& x_2$
Verify if $X'_2 \neq 1$
Verify if $X_2 \neq 1$
$X_5 = (X_1 X'_1 X'_2)^{x_2 s}$
$X'_5 = (X_1 X_2 X'_1)^{x'_2 s}$
Knowledge proofs for $x_2 s$
Knowledge proofs for $x'_2 s$
$X_5, \mathrm{KP}\{x_2 s\}$
$X'_5, \mathrm{KP}\{x'_2 s\}$
$X'_5, \mathrm{KP}\{x'_2 s\}$
$X_5, \mathrm{KP}\{x_2 s\}$
Verify $\mathrm{KP}\{x'_2 s\}$
Verify $\mathrm{KP}\{x_2 s\}$
$k = H(g^{(x_1+x'_1)x_2 x'_2 s})$
$k = H(g^{(x_1+x'_1)x_2 x'_2 s})$

Fig. 3: *A scenario for the key-replication attack on the J-PAKE protocol*

4.3 Further Defects

In this section, we consider some extra defects that can be valid according to practical scenarios and stronger security models. As the J-PAKE protocol was not accompanied by formal proofs of security, we may consider its security according to different security models. As a two-party protocol with password-only authentication, we can not expect the J-PAKE protocol to provide those security guarantees that are provided by multi-factor authentication PAKE protocols or even those two-party PAKE protocols that require the server to have a public key. However, as the J-PAKE protocol has been recommended as a replacement to PKI-based protocols, and has already been used in some scenarios where the server can have a public key, it is noteworthy to discuss further security problems according to stronger security models.

4.3.1 DoS attack

The first round of J-PAKE does not use any pre-shared secret, e.g. password. Then, any entity 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 [18, 28]. From the DoS attack perspective, it would be better to have knowledge proofs of password at the first round. Then, an attacker who did not share any password with the server, would be detected at the beginning.

4.3.2 Server compromise

As mentioned in [8, 9], J-PAKE is a *balanced* protocol, and is vulnerable to the *server compromise*. By *balanced* PAKE protocols, the authors 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 the J-PAKE protocol to server compromise is justified by challenging the need for *augmented* PAKE protocols [8, 9]: (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.

The above argument would be 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* [17] 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. For example, Martini [11] showed how improper implementation of the J-PAKE protocol in OpenSSL and OpenSSH makes it vulnerable to attacks, while those attacks are not applicable to the J-PAKE protocol in theory.

4.3.3 Vulnerability to bad randomness

Security models for capturing bad randomness in AKE protocols is a recent and interesting topic for research. Bad randomness can happen when an adversary compromises the randomness source and hence directly controls the randomness of each AKE session, or when the randomness repeats in different sessions due to reset attacks [29]. As the J-PAKE protocol has been recommended for inclusion in the standards, it would be nice to study its behaviour under bad randomness.

If ZKPs are implemented using the Schnorr signature as suggested in [8, 9], for knowledge proof of x_2s, $\mathcal{A}$ sends $r_5 = v_5 - sx_2h_5$ where r_5 and h_5 are publicly known. For knowledge proof of x_4s, $\mathcal{B}$ sends $r_6 = v_6 - sx_4h_6$ in which r_6 and h_6 are publicly known. Then, the practical security of the J-PAKE protocol 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 extracted.

4.3.4 Password Compromise Impersonation attack

A main advantage of the J-PAKE protocol, as mentioned in [8, 9], is that it does not require the server to have a public key and then, there is no need for a PKI. However, there are notions of security that cannot be provided by a protocol that is solely based on a pre-shared low-entropy password. Resilience to the password compromise impersonation (PCI) attack is a stronger notion of security that can be provided by some PAKE protocols that include more authentication factors than only a pre-shared password [30]. Upon disclosure of $\mathcal{A}$'s password, $\mathcal{M}$ can impersonate $\mathcal{A}$ and share a session key with $\mathcal{B}$, but it should not enable $\mathcal{M}$ to impersonate $\mathcal{B}$ in communication with $\mathcal{A}$. This notion obviously cannot be provided by a two-party password-only PAKE protocol where the only secret parameter is a password. Security of traditional two-party password-only PAKE protocols is typically proven in the BPR model [7]. Once the password is compromised, the security will be jeopardized. Results from the BPR model is disputed in the literature [2, 13]. Regardless, the assumption for password security is not so realistic. In practice, people use the same and often weak passwords for different applications. Upon compromising a client's password in a password-only PAKE protocol, an adversary can impersonate the server, and establish an authenticated session key with the client. The client would then send her sensitive information, e.g. her credit card number to the adversary. Thus, it is nice to have protocols that are resilient to the PCI attack [16, 31], although such protocols will not be solely based on a password [30].

It is trivial to show that the J-PAKE protocol is vulnerable to the PCI attack. Upon disclosure of client $\mathcal{A}$'s password, $\mathcal{M}$ can masquerade as $\mathcal{B}$, and share an authenticated session key with $\mathcal{A}$.

4.3.5 Computational costs

The J-PAKE protocol requires many computations that includes at least 28 exponentiations and 10 random number generations if the Schnorr signature is used for zero-knowledge proofs. However, it was claimed for its efficiency in [8, 9].

5 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 [12]. 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 [32]. 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. 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 [15]. It is noteworthy that any web server that supports TLS/SSL protocol, has already 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.

6 Conclusion

Security of the J-PAKE protocol [8, 9] was analyzed in this paper. We showed that the J-PAKE protocol is vulnerable to the UKS attack if unique identifiers are not included in the zero-knowledge proofs. The J-PAKE protocol is vulnerable to the key-replication attack if the server is allowed to be the initiator. However, it is possible to avoid UKS and key-replication attacks by modifying the key derivation function. The J-PAKE protocol requires at least 28 exponentiations and 10 random number generations if zero-knowledge proofs are implemented by the Schnorr signature. This can make it computationally costly compared to some PAKE protocols. As a two-party PAKE protocol with password-only authentication, the J-PAKE protocol is trivially vulnerable to the password compromise impersonation attack. As the password-based authentication is postponed to end of the J-PAKE protocol, it can increase server's vulnerability to the DoS attack. The J-PAKE protocol is also trivially vulnerable to the server compromise. Application of the J-PAKE protocol 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] Bellovin, S., Merritt, M.: 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] Choo, K.-K. R., Boyd, C., Hitchcock, Y.: Examining indistinguishability-based proof models for key establishment protocols. In *Advances in Cryptology - ASIACRYPT 2005*, vol. 3788 of *Lecture Notes in Computer Science*, pp. 585–604. Springer Berlin Heidelberg, 2005.

[3] Wang, W., Hu, L.: Efficient and provably secure generic construction of three-party password-based authenticated key exchange protocols. In *Progress in Cryptology - INDOCRYPT 2006*, vol. 4329 of *Lecture Notes in Computer Science*, pp. 118–132. Springer Berlin Heidelberg, 2006.

[4] Byun, J. W., Lee, D. H., Lim, J. I.: Security analysis and improvement of a gateway-oriented password-based authenticated key exchange protocol. *Communications Letters, IEEE* 10(9), 683–685, Sept 2006.

[5] Wei, F., Ma, C., Zhang, Z.: Gateway-oriented password-authenticated key exchange protocol with stronger security. In *Provable Security*, vol. 6980 of *Lecture Notes in Computer Science*, pp. 366–379. Springer Berlin Heidelberg, 2011.

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

[7] Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology–EUROCRYPT'00*, vol. 1807 of *Lecture Notes in Computer Science*, pp. 139–155. Springer Berlin Heidelberg, 2000.

[8] Hao, F., Ryan, P.: Password authenticated key exchange by juggling. In *16th International Workshop on Security Protocols 2008*, vol. 6615 of *Lecture Notes in Computer Science*, pp. 159–171. Springer

Berlin Heidelberg, 2011.

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

[10] IEEE P1363.2: Password-based public-key cryptography. http://grouper.ieee.org/groups/1363/passwdPK/submissions.html.

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

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

[13] Zhao, Z., Dong, Z., Wang, Y.: Security analysis of a password-based authentication protocol proposed to IEEE 1363. *Theoretical Computer Science* 352(1), 280–287, 2006.

[14] Yang, J., Cao, T.: Provably Secure Three-party Password Authenticated Key Exchange Protocol in the Standard Model. *Journal of Systems and Software* 85(2), 340–350, 2012.

[15] Halevi, S., Krawczyk, H.: Public-key cryptography and password protocols. *ACM Transactions on Information and System Security (TISSEC)* 2(3), 230–268, 1999.

[16] Feng, D.-G., Xu, J.: A new client-to-client password-authenticated key agreement protocol. In *Second International Workshop on Coding and Cryptology (IWCC'09)*, vol. 5557 of *Lecture Notes in Computer Science*, pp. 63–76. Springer Berlin Heidelberg, 2009.

[17] Gentry, C., MacKenzie, P., Ramzan, Z.: A method for making password-based key exchange resilient to server compromise. In *Advances in Cryptology - CRYPTO 2006*, vol. 4117 of *Lecture Notes in Computer Science*, pp. 142–159. Springer Berlin Heidelberg, 2006.

[18] Boyd, C., Gonzalez-Nieto, J., Kuppusamy, L., Narasimhan, H., Rangan, C., Rangasamy, J., Smith, J., Stebila, D.,

Varadarajan, V.: Cryptographic approaches to denial-of-service resistance. In *An Investigation into the Detection and Mitigation of Denial of Service (DoS) Attacks*, pp. 183–238. Springer India, 2011.

[19] Schnorr, C.: Efficient signature generation by smart cards. *Journal of Cryptology* 4(3), 161–174, 1991.

[20] Toorani, M., Beheshti, A.: 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.

[21] Toorani, M.: SMEmail - a new protocol for the secure e-mail in mobile environments. In *Proceedings of the Australian Telecommunications Networks and Applications Conference (ATNAC'08)*, pp. 39–44. IEEE, 2008.

[22] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology - CRYPTO'86*, vol. 263 of *Lecture Notes in Computer Science*, pp. 186–194. Springer Berlin Heidelberg, 1987.

[23] Toorani, M., Beheshti, A.: Cryptanalysis of an elliptic curve-based signcryption scheme. *International Journal of Network Security* 10(1), 51–56, 2010.

[24] Tang, Q., Mitchell, C. J.: On the security of some password-based key agreement schemes. In *International Conference on Computational Intelligence and Security (CIS'05)*, vol. 3802 of *Lecture Notes in Computer Science*, pp. 149–154. Springer Berlin Heidelberg, 2005.

[25] Implementation of J-PAKE in python. `https://github.com/warner/python-jpake`. Accessed: 2014-08-21.

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

[27] Choo, K.-K. R., Boyd, C., Hitchcock, Y.: 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.

[28] Meadows, C.: A formal framework and evaluation method for network denial of service. In *Proceedings of the 12th IEEE Workshop on Computer Security Foundations (CSFW'99)*, pp. 4–13. IEEE Computer Society, Washington, DC, USA, 1999.

[29] Yang, G., Duan, S., Wong, D., Tan, C., Wang, H.: Authenticated key exchange under bad randomness. In *Proceedings of the 15th International Conference on Financial Cryptography and Data Security (FC'11)*, vol. 7035 of *Lecture Notes in Computer Science*, pp. 113–126. Springer Berlin Heidelberg, 2012.

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

[31] Xiao-fei, D., Chuan-gui, M., Qing-feng, C.: Password authenticated key exchange protocol with stronger security. In *First International Workshop on Education Technology and Computer Science (ETCS'09)*, vol. 2, pp. 678–681. March 2009.

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