# Security Analysis of J-PAKE

Mohsen Toorani
Department of Informatics, University of Bergen
P.O.Box 7803, N-5020 Bergen, Norway
mohsen.toorani@ii.uib.no

**Abstract**

J-PAKE is a balanced Password-Authenticated Key Exchange (PAKE) protocol, proposed in 2008 and presented again in 2010 and 2011. One of its distinguishing features is that it does not require Public Key Infrastructure (PKI). Instead, it deploys Zero-Knowledge (ZK) techniques through the Schnorr's signature, and requires many computations and random number generations. J-PAKE has been submitted as a candidate for the IEEE P1363.2 standard for password-based public key cryptography, included in OpenSSL and OpenSSH, and used in the Mozilla Firefox's Sync mechanism. In this paper, we show that the J-PAKE protocol is vulnerable to a password compromise impersonation attack, and has other shortcomings with respect to replay and Unknown Key-Share (UKS) attacks.

## 1 Introduction

Password-Authenticated Key Exchange (PAKE) protocols enable entities to authenticate each other and share a strong cryptographic key based on a pre-shared human memorable password. Although it seemed incredible to construct a strong cryptographic session key from a low-entropy password, Bellowin and Meritt [1] were the first researchers that demonstrated the PAKE problem is solvable. They introduced the first PAKE protocol, called EKE, in 1992. Their protocol was shown to have some weaknesses [2–5]. Since the introduction of EKE, many PAKE protocols have been proposed. Some of them were proved to have security problems. With introduction of new technologies and fast improvements in computational capabilities of entities, including honest users and attackers, new notions of security should be defined and new protocols are required for different paradigms. Computational and communication efficiency are also two factors that are considered in designing new protocols.

Password-Authentication Key Exchange by Juggling (J-PAKE) is a PAKE protocol that was initially proposed in [6] and presented again in [7, 8]. Since 2008, it has been available on the website of the IEEE P1363.2 project for standard specifications of password-based public-key cryptography [9] but as J-PAKE's designers argue [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]. J-PAKE has also been used for the Sync mechanism in Mozilla Firefox 4 (beta 8 and later) [10]. J-PAKE's designers argue that it is a balanced (opposed to an augmented) PAKE protocol [6–8, 10]. Balanced PAKE protocols allow parties to use the same password, and are supposed to be vulnerable to the server compromise. In augmented PAKE protocols that are more customized for the client-server case, however, an attacker must perform a successful brute-force attack in order to masquerade as the client using stolen server data. The J-PAKE is balanced and its designers justified its vulnerability to the server compromise by challenging the necessity of augmented PAKE protocols as they argue [6–8]: (1) No of the 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.

This kind of argument is disputable as: (1) Even if some augmented PAKE protocols are vulnerable to an offline dictionary attack after the server compromise, it usually requires a separate offline dictionary attack for each entry or user. Then, it would require a lot of time to extract passwords of all users, and users may be notified of the server compromise during this process and change their passwords. The situation is different for balanced protocols where all passwords are supposed to be compromised immediately by the server compromise. (2) The second reasoning is obviously against what we expect from a protocol to express all stages clearly and consider the effect of any manipulation on the security and computational costs. All the security considerations should be considered in protocol design and protocols should be implemented without any modification as it can cause security problems and make the protocol vulnerable to attacks. Explicitly, Martini [11] showed how improper implementation of J-PAKE in OpenSSL and OpenSSH makes it vulnerable to attacks while those attacks are not applicable to the J-PAKE protocol in theory.

Resistance to offline and online dictionary attacks, forward secrecy, and known session key security are four security requirements that are mentioned in [6–8] as the security requirements of a PAKE protocol and it is claimed that J-PAKE provides all of them. In this paper, it is shown that the J-PAKE protocol is vulnerable to a password compromise impersonation attack, and has some other shortcomings regarding replay and UKS attacks. Actually, the security requirements of PAKE protocols are not confined to the four aforementioned properties. There are other security attributes that are desired or required for a PAKE protocol that are briefly described in Section 2. Section 3 briefly reviews the J-PAKE protocol, and Section 4 explains its security vulnerabilities. Finally, Section 5 concludes the paper.

## 2 Security Requirements

There are some security attributes that PAKE protocols should possess [12–14]. J-PAKE's designers argue that it is a balanced PAKE protocol and does not have any resilience to the server compromise. J-PAKE does not use certified public keys so it should not provide some security attributes such as resilience to Key Compromise Impersonation (KCI) attack, etc. Instead, it should provide the following security attributes:

- **Resilience to offline dictionary attack:** A PAKE protocol should not reveal any information that can be used as a verifier for an offline 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. He applies each password from the dictionary to the obtained verifier until he obtains the correct password that satisfies the verifier. Resilience to offline dictionary attack is a major security attribute for PAKE protocols.

- **Resilience to online dictionary attack:** In an online dictionary attack, the adversary uses a dictionary of passwords but checks the validity of his guess through online transactions with the target. Unlike an offline dictionary attack that is a passive attack, an online dictionary attack is an active attack. For preventing this attack, servers usually lock the account of the corresponding user after several number of 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 in order to obtain information and uses it for doing an offline dictionary attack. As checking passwords are accomplished offline, the server will not detect any attack. Resilience to (undetectable) online dictionary attack is also a major security attribute for PAKE protocols.

- **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.

- **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.

- **No 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.

- **Resilience to Unknown Key-Share (UKS) 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 $\mathcal{E}$ and $\mathcal{B}$ correctly believes the key is shared with $\mathcal{A}$.

- **Resilience to Denning-Sacco attack:** It prevents an adversary to recover or guess the password upon disclosure of a session key.
- **Resilience to replay attack:** In the 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 password compromise impersonation attack:** When the password of entity $\mathcal{A}$ is disclosed, adversary $\mathcal{E}$ that has $\mathcal{A}$'s password can impersonate $\mathcal{A}$ but it should not enable $\mathcal{E}$ to impersonate another honest entity and share a session key with $\mathcal{A}$. It is an important attribute for PAKE protocols and they should be resilient to a password compromise impersonation attack.
- **Resilience to ephemeral key compromise impersonation attack:** PAKE protocols deploy some random numbers as ephemeral keys. Disclosure of an ephemeral key of any entity $\mathcal{A}$ should not enable an adversary to share a session key with $\mathcal{A}$ by impersonating another entity.
- **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.
- **Resilience to malicious server attack:** The malicious server attack has been considered in some papers [15–18], and resilience to this attack has been mentioned as a security requirement of password-based protocols that thwart the phishing attacks. However, it is not a strict security requirement, and it is disputable. In a malicious server attack, an adversary runs on a malicious server and tempts people to register with that server. As people usually use the same password for login into different servers, the malicious server may be able to use this password to impersonate the user and login into other honest servers.

## 3 Review of the J-PAKE protocol

Figure 1 depicts the top-level description of the J-PAKE protocol as presented in [6–8]. J-PAKE requires four passes of communication between two communicating entities, Alice and Bob, but the protocol can be completed in two rounds. In the rest of this paper, Alice, Bob and the adversary will be denoted by $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{E}$, respectively.

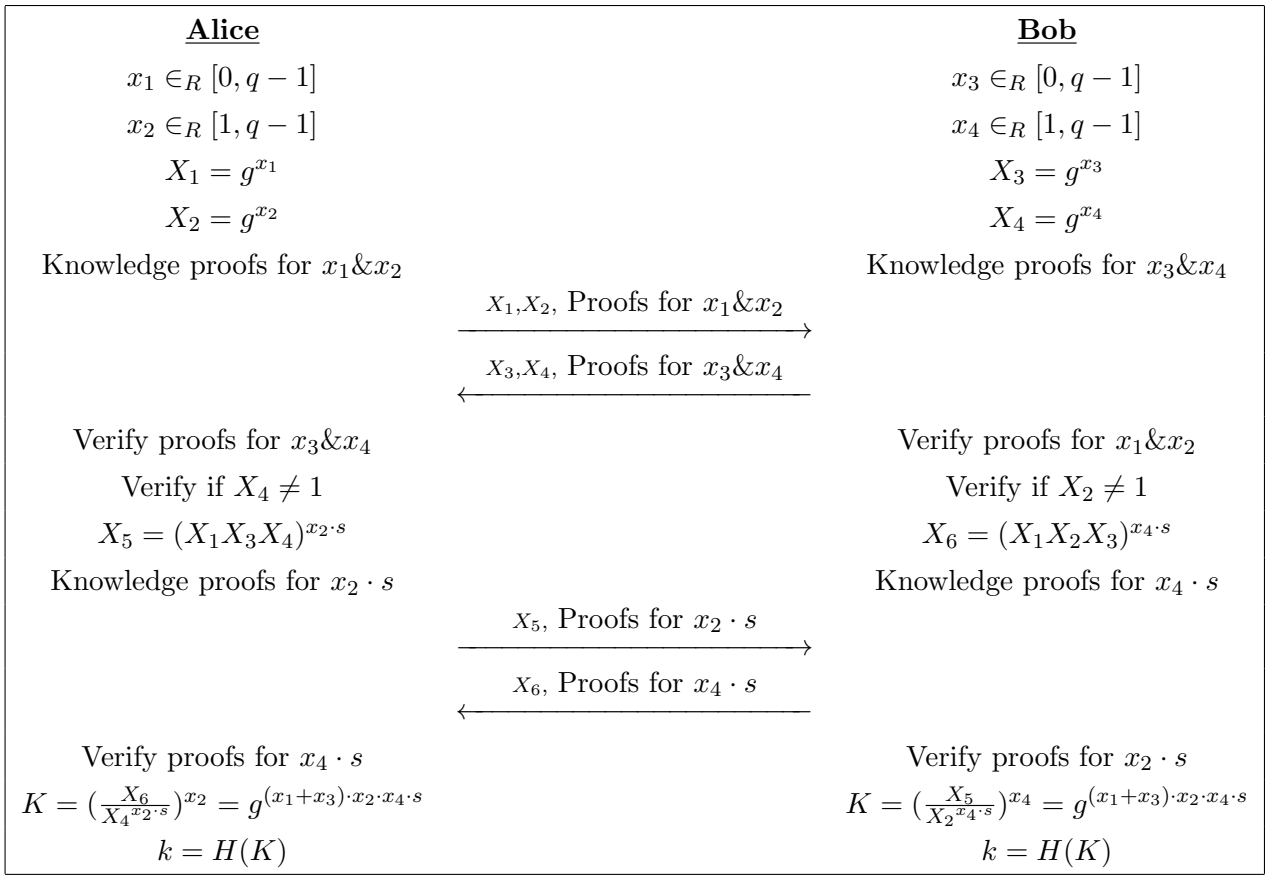


Figure 1: Top-level description of the J-PAKE protocol

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 indicated in [8], $s$ may also be a hash of the shared password together with some salt. Steps for top-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 the 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 protocols run.

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 the 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 protocols run.

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 protocols run.

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 protocols run.

By completion of successful verifications, $\mathcal{A}$ and $\mathcal{B}$ authenticate each other and obtain the same session key $k$. The correctness can be simply verified as $K = (\frac{X_6}{X_4^{x_2 \cdot s}})^{x_2} = (\frac{X_5}{X_2^{x_4 \cdot s}})^{x_4} = g^{(x_1+x_3) \cdot x_2 \cdot x_4 \cdot s}$. As 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 top-level description of the J-PAKE protocol, depicted in Figure 1 [6–8], is misleading and ambiguous. It includes fundamental blocks for Zero-Knowledge Proofs (ZKP) and verifications without further details. However, J-PAKE's specifications [6–8] suggest using Schnorr's signature [19] for ZKP, and take care of it in comparing the computational costs, etc. However, they do not

provide a clear description of the protocol, and descriptions are ambiguous. 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 $H$. To prove the 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.

# 4 Security Problems of J-PAKE

In [6–8], J-PAKE is claimed to be resilient to online and offline dictionary attacks. However, there are no strict 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, it is shown that J-PAKE lacks some of the security attributes explained in Section 2.

## 4.1 Vulnerability to a Password Compromise Impersonation attack

PAKE protocols should be resilient to the password compromise impersonation attack. That is, even with compromise of $\mathcal{A}$'s low-entropy password $s$, $\mathcal{E}$ should not be able to impersonate $\mathcal{B}$ and share a session key with $\mathcal{A}$. J-PAKE does not provide this attribute and is vulnerable to this attack. $\mathcal{E}$ can have control on the communication link of $\mathcal{A}$. Here is the attack scenario:

- $\mathcal{A}$ or $\mathcal{E}$ start the protocol. $\mathcal{A}$ selects $x_1$ and $x_2$. $\mathcal{A}$ computes $\{X_1,\ X_2,\ ZKP\ for\ x_1 \& x_2\}$, and sends them to $\mathcal{E}$. $\mathcal{E}$ selects $x_3$ and $x_4$. $\mathcal{E}$ computes $\{X_3,\ X_4,\ ZKP\ for\ x_3 \& x_4\}$, 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 $\mathcal{E}$. $\mathcal{E}$ computes $\{X_6$ and ZKP for $x_4 \cdot s\}$, and sends them to $\mathcal{A}$. As $\mathcal{E}$ 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 $k$. $\mathcal{E}$ just computes $K$ and $k$ as he does not need to authenticate $\mathcal{A}$. $\mathcal{E}$ could successfully impersonate $\mathcal{B}$ and share a session key with $\mathcal{A}$.

This is a password compromise impersonation attack on the J-PAKE protocol.

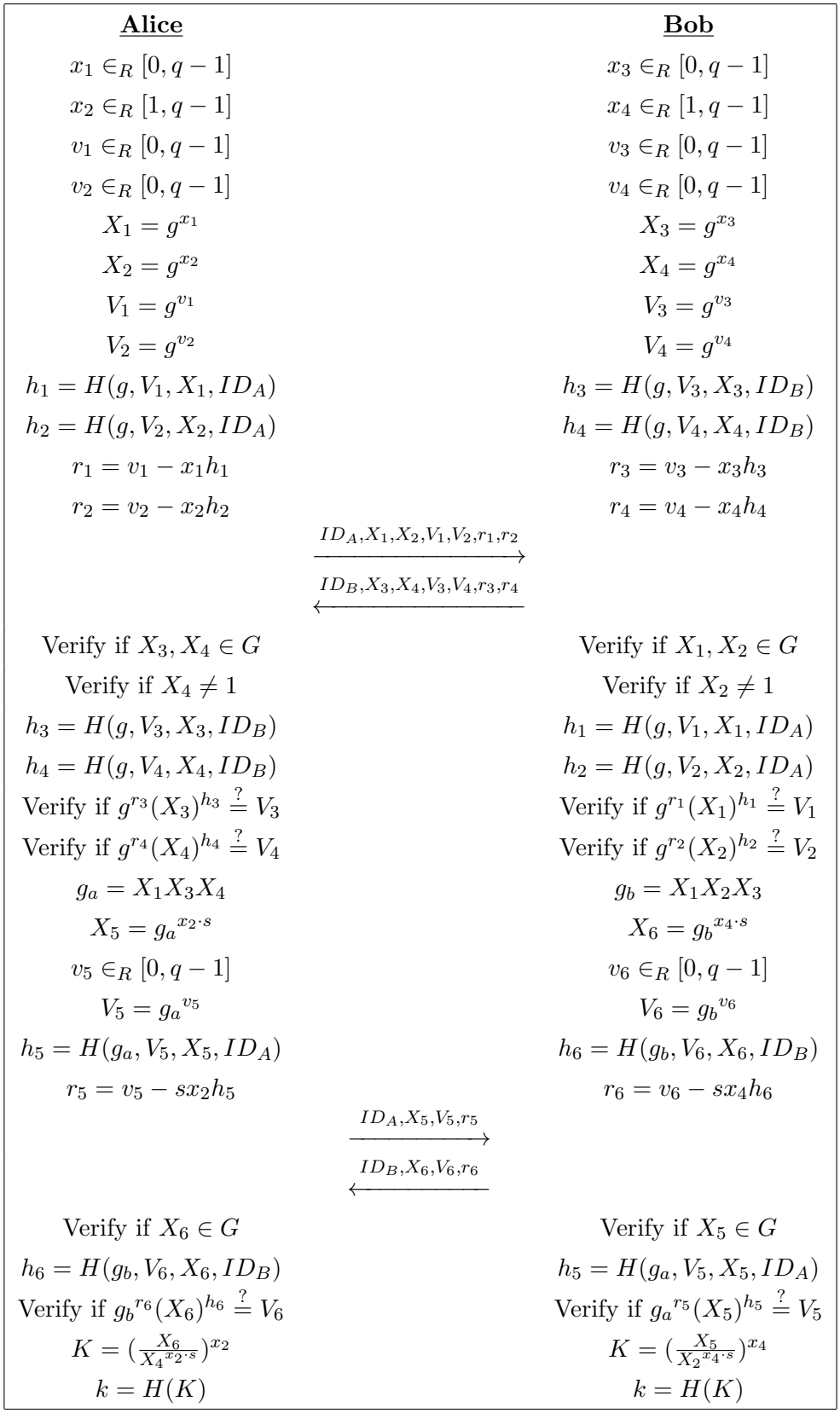


Figure 2: Detailed description of the J-PAKE protocol after substituting ZKP with Schnorr's signature as proposed in [6–8]

### 4.2 Vulnerability to a replay attack

J-PAKE has two 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 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.

These features can be misused through a replay attack. This means that the adversary $\mathcal{E}$ can successfully thwart all the authentications by replaying previous messages, eavesdropped from previous runs of the protocol between $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{E}$ 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 $\mathcal{E}$ as $\mathcal{A}$ or $\mathcal{B}$, respectively. Of course, the adversary cannot compute the session key but all the authentications will be successful. There may be some scenarios that this attack would be useful (e.g. a naive applicant may suppose that the J-PAKE is a strong protocol and use it just for the authentication and ignore the key generation part) but it exhibits some weaknesses in the J-PAKE protocol. Assuming that $\mathcal{E}$ impersonates $\mathcal{A}$, steps for this attack can be followed as:

1. $\mathcal{E}$ 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 $\mathcal{E}$.

3. $\mathcal{E}$ 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 $h_1'$ and $h_2'$, 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 $\mathcal{E}$. $\mathcal{B}$ 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 $\mathcal{E}$ 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. $\mathcal{E}$ does not have $s$ and $x_2'$ so $\mathcal{E}$ cannot compute the session key but could successfully impersonate $\mathcal{A}$.

Similar steps can be written for the case when $\mathcal{E}$ impersonates $\mathcal{B}$.

### 4.3 Further Defects

1. 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* if there is not any binding between identifiers and authentication credentials in the J-PAKE protocol. The top-level 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 the authentication credentials and the session key derivation function. Fortunately, in our detailed description of the J-PAKE that is shown 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, without specific modifications for binding ZKP with identifiers of $\mathcal{A}$ and $\mathcal{B}$ as described in Figure 2, it may 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 $k = H(K, ID_A, ID_B)$.

2. As J-PAKE's designers explicitly argue [8], J-PAKE is a balanced protocol. It is then vulnerable to the server compromise and *Stolen-verifier attacks*.

3. The J-PAKE protocol is also vulnerable to the *malicious server attack*. However, as invulnerability to this attack is not a strict security requirement of PAKE protocols and it is usually considered as a security requirement for anti-phishing PAKE protocols, we do not consider it in details.

4. The J-PAKE protocol requires many computations which includes 28 exponentiations and 10 random number generations. The first round of the J-PAKE does not use any pre-shared secret, e.g. password, so anyone can successfully complete the first round. The second round also includes some weaknesses that makes the protocol vulnerable to different attacks described in this section. Although J-PAKE is not based on certified public keys and PKI, it would not be secure and efficient enough to be considered as an alternative for PKI-based protocols.

5. It is better to exclude zero from the allowed values for random numbers $\{x_1, x_3, v_1, v_2, v_3, v_4, v_5, v_6\}$. Specifically, for $x_1 = x_3 = 0$, we have $K = 1$.

## 5 Conclusion

The security of the J-PAKE protocol [6–8] as a balanced PAKE protocol was analyzed in this paper. Although J-PAKE's designers argue that it is secure, it was shown that it has some

security weaknesses that expose it to some attacks. It includes its vulnerability to Password Compromise Impersonation and replay attacks. It was also claimed for computational efficiency of the J-PAKE when it was partially compared with SPEKE and EKE protocols, but J-PAKE involves many random number generations and mathematical manipulations that render it an inefficient protocol, especially for resource-constrained environments.

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

[3] B. Jaspan, "Dual-workfactor Encrypted Key Exchange: Efficiently Preventing Password Chaining and Dictionary Attacks," in *Proceedings of the 6th Annual USENIX Security Conference*, pp. 43–50, 1996.

[4] 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.

[5] P. MacKenzie, "The PAK suite: Protocols for Password-Authenticated Key Exchange." IEEE P1363.2, 2002.

[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 Workshop on Security Protocols, 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 (SPW)*, (Cambridge, UK), 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] 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.

[13] H. Yeh and H. Sun, "Password Authenticated Key Exchange Protocols among diverse network domains," *Computers and Electrical Engineering*, vol. 31, no. 3, pp. 175–189, 2005.

[14] P. Nose, "Security Weaknesses of Authenticated Key Agreement Protocols," *Information Processing Letters*, vol. 111, no. 14, pp. 687–696, 2011.

[15] S. Sood, A. Sarje, and K. Singh, "A secure dynamic identity based authentication protocol for multi-server architecture," *Journal of Network and Computer Applications*, vol. 34, no. 2, pp. 609–618, 2011.

[16] M. Gouda, A. Liu, L. Leung, and M. Alam, "SPP: An anti-phishing single password protocol," *Computer Networks*, vol. 51, no. 13, pp. 3715–3726, 2007.

[17] J. Byun, D. Lee, and J. Lim, "EC2C-PAKA: An efficient client-to-client password-authenticated key agreement," *Information Sciences*, vol. 177, no. 19, pp. 3995–4013, 2007.

[18] D. Wong, "Security analysis of two anonymous authentication protocols for distributed wireless networks," in *Third IEEE International Conference on Pervasive Computing and Communications (PerCom'05) Workshops*, pp. 284–288, IEEE, 2005.

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