# J-PAKE: Authenticated Key Exchange without PKI

Feng Hao[1] and Peter Ryan[2]

[1] Thales E-Security, Cambridge, UK
[2] Faculty Of Science, University of Luxembourg

**Abstract.** Password Authenticated Key Exchange (PAKE) is one of the important topics in cryptography. It aims to address a practical security problem: how to establish secure communication between two parties solely based on a shared password without requiring a Public Key Infrastructure (PKI). After more than a decade of extensive research in this field, there have been several PAKE protocols available. The EKE and SPEKE schemes are perhaps the two most notable examples. Both techniques are however patented. In this paper, we review these techniques in detail and summarize various theoretical and practical weaknesses. In addition, we present a new PAKE solution called J-PAKE. Our strategy is to depend on well-established primitives such as the Zero-Knowledge Proof (ZKP). So far, almost all of the past solutions have avoided using ZKP for the concern on efficiency. We demonstrate how to effectively integrate the ZKP into the protocol design and meanwhile achieve good efficiency. Our protocol has comparable computational efficiency to the EKE and SPEKE schemes with clear advantages on security.

**Keywords:** Password-Authenticated Key Exchange, EKE, SPEKE, key agreement.

## 1 Introduction

Nowadays, the use of passwords is ubiquitous. From on-line banking to accessing personal emails, the username/password paradigm is by far the most commonly used authentication mechanism. Alternative authentication factors, including tokens and biometrics, require additional hardware, which is often considered too expensive for an application.

However, the security of a password is limited by its low-entropy. Typically, even a carefully chosen password only has about 20-30 bits entropy [3]. This makes passwords subject to dictionary attacks or simple exhaustive search. Some systems willfully force users to remember cryptographically strong passwords, but that often creates more problems than it solves [3].

Since passwords are weak secrets, they must be protected during transmission. Currently, the widely deployed method is to send passwords through SSL/TLS [29]. But, this requires a Public Key Infrastructure (PKI) in place; maintaining a PKI is expensive. In addition, using SSL/TLS is subject to man-in-the-middle

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. XI, LNCS 6480, pp. 192–206, 2010.

attacks [3]. If a user authenticates himself to a phishing website by disclosing his password, the password will be stolen even though the session is fully encrypted.

The PAKE research explores an alternative approach to protect passwords without relying on a Public Key Infrastructure (PKI) at all [10, 16]. It aims to achieve two goals. First, it allows zero-knowledge proof of the password. One can prove the knowledge of the password without revealing it to the other party. Second, it performs authenticated key exchange. If the password is correct, both parties will be able to establish a common session key that no one else can compute.

The first milestone in PAKE research came in 1992 when Bellovin and Merrit introduced the Encrypted Key Exchange (EKE) protocol [10]. Despite some reported weaknesses [16, 20, 23, 25], the EKE protocol first demonstrated that the PAKE problem was at least solvable. Since then, a number of protocols have been proposed. Many of them are simply variants of EKE, instantiating the "symmetric cipher" in various ways [7].

The few techniques that claim to resist known attacks have almost all been patented. Most notably, EKE was patented by Lucent Technologies [12], SPEKE by Phoenix Technologies [18] and SRP by Stanford University [28]. The patent issue is arguably one of the biggest brakes in deploying a PAKE solution in practice [13].

## 2 Past Work

### 2.1 Security Requirements

Before reviewing past solutions in detail, we summarize the security requirements that a PAKE protocol shall fulfill (also see [10, 11, 16, 28]).

1. **Off-line dictionary attack resistance** – It does not leak any information that allows a passive/active attacker to perform off-line exhaustive search of the password.
2. **Forward secrecy** – It produces session keys that remain secure even when the password is later disclosed.
3. **Known-session security** – It prevents a disclosed session from affecting the security of other established session keys.
4. **On-line dictionary attack resistance** – It limits an active attacker to test only one password per protocol execution.

First, a PAKE protocol must resist off-line dictionary attacks. An attacker may be passive (only eavesdropping) or active (directly engaging in the key exchange). In either case, the communication must not reveal any data – say a hash of the password – that allows an attacker to learn the password through off-line exhaustive search.

Second, the protocol must be forward-secure. The key exchange is authenticated based on a shared password. However, there is no guarantee on the long-term secrecy of the password. A well-designed PAKE scheme should protect past session keys even when the password is later disclosed. This property also implies that if an attacker knows the password but only passively observes the key exchange, he cannot learn the session key.

Third, the protocol must provide known session security. If an attacker is able to compromise a session, we assume he can learn all session-specific secrets. However, the impact should be minimized such that a compromised session must not affect the security of other established sessions.

Finally, the protocol must resist on-line dictionary attacks. If the attacker is directly engaging in the key exchange, there is no way to prevent such an attacker trying a random guess of the password. However, a secure PAKE scheme should mitigate the effect of the on-line attack to the minimum – in the best case, the attacker can only guess exactly one password per impersonation attempt. Consecutively failed attempts can be easily detected and thwarted accordingly.

Some papers add an extra "server compromise resistance" requirement: an attacker should not be able to impersonate users to a server after he has stolen the password verification files stored on that server, but has not performed dictionary attacks to recover the passwords [7, 17, 28]. Protocols designed with this additional requirement are known as the augmented PAKE, as opposed to the balanced PAKE that does not have this requirement.

However, the so-called "server compromise resistance" is disputable [24]. First, one may ask whether the threat of impersonating users to a *compromised* server is significantly realistic. After all, the server had been compromised and the stored password files had been stolen. Second, none of the augmented schemes can provide any real assurance once the server is indeed compromised. If the password verification files are stolen, off-line exhaustive search attacks are inevitable. All passwords will need to be revoked and updated anyway.

Another argument in favor of the augmented PAKE is that the server does not store a plaintext password so it is more secure than the balanced PAKE [28]. This is a misunderstanding. The EKE and SPEKE protocols are two examples of the balanced PAKE. Though the original EKE and SPEKE papers only mention the use the plaintext password as the shared secret between the client and server [10, 16], it is trivial to use a hash of the password (possibly with some salt) as the shared secret if needed. So, the augmented PAKE has no advantage in this aspect.

Overall, the claimed advantages of an augmented PAKE over a balanced one are doubtful. On the other hand, the disadvantages are notable. With the added "server compromise resistance" requirement that none of the augmented PAKE schemes truly satisfy [7, 17, 28], an augmented PAKE protocol is significantly more complex and more computationally expensive. The extra complexity opens more opportunities to the attacker, as many of the attacks are applicable on the augmented PAKE [7].

### 2.2 Review on EKE and SPEKE

In this section, we review the two perhaps most well-known balanced PAKE protocols: EKE [10] and SPEKE [16]. Both techniques are patented and have been deployed in commercial applications.

There are many other PAKE protocols in the past literature [7]. Due to the space constraint, we can only briefly highlight some of them. Goldreich and Lindell first provided a formal analysis of PAKE, and they also presented a PAKE protocol that satisfies the formal definitions [33]. However, the Goldreich-Lindell protocol is based on generic multi-party secure computation; it is commonly seen as too inefficient for practical use [34, 35]. Later, there are Abdalla-Pointcheval [1], Katz-Ostrovsky-Yung [34], Jiang-Gong [35] and Gennaro-Lindell [39] protocols, which are proven secure in a common reference model (Abdalla-Pointcheval additionally assumes a random oracle model [1]). All these protocols require a "trusted third party" to define the public parameters: more specifically, the security of the protocol relies on the "independence" of two group generators selected honestly by a trusted third party [1, 34, 35][1]. Thus, as with any "trusted third party", the party becomes the one who can break the protocol security [3]. (Recall that the very goal of PAKE is to establish key exchange between two parties without depending on any external trusted party.) Another well-known provably secure PAKE is a variant of the EKE protocol with formal security proofs due to Bellare, Pointcheval and Rogaway [5] (though the proofs are disputed in [7, 32], as we will explain later). In general, all the above protocols [33, 34, 35, 1, 39, 5] are significantly more complex and less efficient than the EKE and SPEKE protocols. In this paper, we will focus on comparing our technique to the EKE and SPEKE protocols.

First, let us look at the EKE. Bellovin and Merrit introduced two EKE constructs: based on RSA (which was later shown insecure [23]) and Diffie-Hellman (DH). Here, we only describe the latter, which modifies a basic DH protocol by symmetrically encrypting the exchanged items. Let $\alpha$ be a primitive root modulo $p$. In the protocol, Alice sends to Bob $[\alpha^{x_a}]_{\mathrm{s}}$, where $x_a$ is taken randomly from $[1, p-1]$ and $[\ldots]_s$ denotes a symmetric cipher using the password $s$ as the key. Similarly, Bob sends to Alice $[\alpha^{x_b}]_s$, where $x_b \in_R [1, p-1]$. Finally, Alice and Bob compute a common key $K = \alpha^{x_a \cdot x_b}$. More details can be found in [10].

It has been shown that a straightforward implementation of the above protocol is insecure [20]. Since the password is too weak to be used as a normal encryption key, the content within the symmetric cipher must be strictly random. But, for a 1024-bit number modulo $p$, not every bit is random. Hence, a passive attacker can rule out candidate passwords by applying them to decipher $[\alpha^{x_a}]_s$, and then checking whether the results fall within $[p, 2^{1024} - 1]$.

There are suggested countermeasures. In [10], Bellovin and Merrit recommended to transmit $[\alpha^{x_a} + r \cdot p]_s$ instead of $[\alpha^{x_a}]_s$ in the actual implementation, where $r \cdot p$ is added using a non-modular operation. The details on defining $r$ can

[1] The Jiang-Gong paper proposes to use a trusted third party or a threshold scheme to define the public parameters [35], while the KOY paper suggests to use a trusted third party or a source of randomness [34]. However, neither paper provides concrete descriptions of the "threshold scheme" and "source of randomness". The Gennaro-Lindell paper suggests to choose a large organization as the trusted party for all its employees [39]. However, such a setup also severely limits the general deployment of PAKE among the public.

be found in [10]. However, this solution was explained in an ad-hoc way, and it involves changing the existing protocol specification. Due to lack of a complete description of the final protocol, it is difficult to assess its security. Alternatively, Jaspan suggests addressing this issue by choosing $p$ as close to a power of 2 as possible [20]. This might alleviate the issue, but does not resolve it.

The above reported weakness in EKE suggests that formal security proofs are unlikely without introducing new assumptions. Bellare, Pointcheval and Rogaway introduced a formal model based on an "ideal cipher" [5]. They applied this model to formally prove that EKE is "provably secure". However, this result is disputed in [7,32]. The so-called "ideal cipher" was not concretely defined in [5]; it was only later clarified by Boyd et al. in [7]: the assumed cipher works like a random function in encryption, but must map fixed-size strings to elements of $G$ in decryption (also see [32]). Clearly, no such ciphers are readily available yet. Several proposed instantiations of such an "ideal cipher" were easily broken [32].

Another limitation with the EKE protocol is that it does not securely accommodate short exponents. The protocol definition requires $\alpha^{x_a}$ and $\alpha^{x_b}$ be uniformly distributed over the whole group $\mathbb{Z}_p^*$ [10]. Therefore, the secret keys $x_a$ and $x_b$ must be randomly chosen from $[1, p-1]$, and consequently, an EKE must use 1024-bit exponents if the modulus $p$ is chosen 1024-bit. An EKE cannot operate in groups with distinct features, such as a subgroup with prime order – a passive attacker would then be able to trivially uncover the password by checking the order of the decrypted item.

Jablon proposed a different protocol, called Simple Password Exponential Key Exchange (SPEKE), by replacing a fixed generator in the basic Diffie-Hellman protocol with a password-derived variable [16]. In the description of a fully constrained SPEKE, the protocol defines a safe prime $p = 2q + 1$, where $q$ is also a prime. Alice sends to Bob $(s^2)^{x_a}$ where $s$ is the shared password and $x_a \in_R [1, q-1]$; similarly, Bob sends to Alice $(s^2)^{x_b}$ where $x_b \in_R [1, q-1]$. Finally, Alice and Bob compute $K = s^{2 \cdot x_a \cdot x_b}$. The squaring operation on $s$ is to make the protocol work within a subgroup of prime order $q$.

There are however risks of using a password-derived variable as the base, as pointed out by Zhang [31]. Since some passwords are exponentially equivalent, an active attacker may exploit that equivalence to test multiple passwords in one go. This problem is particularly serious if a password is a Personal Identification Numbers (PIN). One countermeasure might be to hash the password before squaring, but that does not resolve the problem. Hashed passwords are still confined to a pre-defined small range. There is no guarantee that an attacker is unable to formulate exponential relationships among hashed passwords; existing hash functions were not designed for that purpose. Hence, at least in theory, this reported weakness disapproves the original claim in [16] that a SPEKE only permits one guess of password in one attempt.

Similar to the case with an EKE, a fully constrained SPEKE uses long exponents. For a 1024-bit modulus $p$, the key space is within $[1, q-1]$, where $q$ is 1023-bit. In [16], Jablon suggested to use 160-bit short exponents in a SPEKE, by choosing $x_a$ and $x_b$ within a dramatically smaller range $[1, 2^{160} - 1]$. But, this would give a passive attacker side information that the $1023 - 160 = 863$ most significant bits in a full-length key are all '0's. The security is not reassuring, as the author later acknowledged in [19].

To sum up, an EKE has the drawback of leaking partial information about the password to a passive attacker. As for a SPEKE, it has the problem that an active attacker may test multiple passwords in one protocol execution. Furthermore, neither protocol accommodates short exponents securely. Finally, neither protocol has security proofs; to prove the security would require introducing new security assumptions [5] or relaxing security requirements [26].

## 3 J-PAKE Protocol

In this section, we present a new balanced PAKE protocol called Password Authenticated Key Exchange by Juggling (J-PAKE). The key exchange is carried out over an unsecured network. In such a network, there is no secrecy in communication, so transmitting a message is essentially no different from broadcasting it to all. Worse, the broadcast is unauthenticated. An attacker can intercept a message, change it at will, and then relay the modified message to the intended recipient.

It is perhaps surprising that we are still able to establish a private and authenticated channel in such a hostile environment solely based on a shared password – in other words, bootstrapping a *high-entropy* cryptographic key from a *low-entropy* secret. The protocol works as follows.

Let $G$ denote a subgroup of $\mathbb{Z}_p^*$ with prime order $q$ in which the Decision Diffie-Hellman problem (DDH) is intractable [6]. Let $g$ be a generator in $G$. The two communicating parties, Alice and Bob, both agree on $(G, g)$. Let $s$ be their shared password[2], and $s \neq 0$ for any non-empty password. We assume the value of $s$ falls within $[1, q-1]$.

Alice selects two secret values $x_1$ and $x_2$ at random: $x_1 \in_R [0, q-1]$ and $x_2 \in_R [1, q-1]$. Similarly, Bob selects $x_3 \in_R [0, q-1]$ and $x_4 \in_R [1, q-1]$. Note that $x_2, x_4 \neq 0$; the reason will be evident in security analysis.

**Round 1.** *Alice sends out $g^{x_1}$, $g^{x_2}$ and knowledge proofs for $x_1$ and $x_2$. Similarly, Bob sends out $g^{x_3}$, $g^{x_4}$ and knowledge proofs for $x_3$ and $x_4$.*

The above communication can be completed in one round as neither party depends on the other. When this round finishes, Alice and Bob verify the received knowledge proofs, and also check $g^{x_2}, g^{x_4} \neq 1$.

**Round 2.** *Alice sends out $\mathcal{A} = g^{(x_1+x_3+x_4)\cdot x_2 \cdot s}$ and a knowledge proof for $x_2 \cdot s$. Similarly, Bob sends out $\mathcal{B} = g^{(x_1+x_2+x_3)\cdot x_4 \cdot s}$ and a knowledge proof for $x_4 \cdot s$.*

When this round finishes, Alice computes $K = (\mathcal{B}/g^{x_2 \cdot x_4 \cdot s})^{x_2} = g^{(x_1+x_3)\cdot x_2 \cdot x_4 \cdot s}$, and Bob computes $K = (\mathcal{A}/g^{x_2 \cdot x_4 \cdot s})^{x_4} = g^{(x_1+x_3)\cdot x_2 \cdot x_4 \cdot s}$. With the same keying material $K$, a session key can be derived $\kappa = H(K)$, where $H$ is a hash function.

[2] Depending on the application, $s$ could also be a hash of the shared password together with some salt.

The two-round J-PAKE protocol can serve as a drop-in replacement for face-to-face key exchange. It is like Alice and Bob meet in person and secretly agree a common key. So far, the authentication is implicit: Alice believes only Bob can derive the same key and vice versa. In some applications, Alice and Bob may want to perform an explicit key confirmation just to make sure the other party actually holds the same key.

There are several ways to achieve explicit key confirmation. In general, it is desirable to use a different key from the session key for key confirmation[3], say use $\kappa' = H(K, 1)$. We summarize a few methods, which are generically applicable to all key exchange schemes. A simple method is to use a hash function similar to the proposal in SPEKE: Alice sends $\mathrm{H}(\mathrm{H}(\kappa'))$ to Bob and Bob replies with $\mathrm{H}(\kappa')$. Another straightforward way is to use $\kappa'$ to encrypt a known value (or random challenge) as presented in EKE. Other approaches make use of MAC functions as suggested in [36]. Given that the underlying functions are secure, these methods do not differ significantly in security.

In the protocol, senders need to produce valid knowledge proofs. The necessity of the knowledge proofs is motivated by Anderson-Needham's sixth principle in designing secure protocols [2]: "*Do not assume that a message you receive has a particular form (such as $g^r$ for known r) unless you can check this.*" Fortunately, Zero-Knowledge Proof (ZKP) is a well-established primitive in cryptography; it allows one to prove his knowledge of a discrete logarithm without revealing it [29].

As one example, we could use Schnorr's signature [30], which is non-interactive, and reveals nothing except the one bit information: "whether the signer knows the discrete logarithm". Let $H$ be a secure hash function[4]. To prove the knowledge of the exponent for $X = g^x$, one sends $\{\text{SignerID}, V = g^v, r = v - xh\}$ where SignerID is the unique user identifier, $v \in_R \mathbb{Z}_q$ and $h = H(g, V, X, \text{SignerID})$. The receiver verifies that $X$ lies in the prime-order subgroup $G$ and that $g^v$ equals $g^r X^h$. Adding the unique SignerID into the hash function is to prevent Alice replaying Bob's signature back to Bob and vice versa. Note that for Schnorr's signature, it takes one exponentiation to generate it and two to verify it (computing $g^r \cdot X^h$ requires roughly one exponentiation using the simultaneous computation technique [37]).

## 4 Security Analysis

In this section, we show the protocol fulfills all the security requirements listed in Section 2.1.

[3] Using a different key has a (subtle) theoretical advantage that the session key will remain indistinguishable from random even after the key confirmation. However, this does not make much difference in practical security and is not adopted in [10, 16].

[4] Schnorr's signature is provably secure in the random oracle model, which requires a secure hash function.

### 4.1 Off-Line Dictionary Attack Resistance

First, we discuss the protocol's resistance against the off-line dictionary attack. Without loss of generality, assume Alice is honest. Her ciphertext $\mathcal{A}$ contains the term $(x_1 + x_3 + x_4)$ on the exponent. Let $x_a = x_1 + x_3 + x_4$. The following lemma shows the security property of $x_a$.

**Lemma 1.** *The $x_a$ is a secret of random value over $\mathbb{Z}_q$ to Bob.*

*Proof.* The value $x_1$ is uniformly distributed over $\mathbb{Z}_q$ and unknown to Bob. The knowledge proofs required in the protocol show that Bob knows $x_3$ and $x_4$. By definition $x_a$ is computed from $x_3$ and $x_4$ (known to Bob) plus a random number $x_1$. Therefore $x_a$ must be randomly distributed over $\mathbb{Z}_q$.

In the second round of the protocol, Alice sends $\mathcal{A} = g_a^{x_2 \cdot s}$ to Bob, where $g_a = g^{x_1+x_3+x_4}$. Here, $g_a$ serves as a generator. As the group $G$ has prime order, any non-identity element is a generator [29]. So Alice can explicitly check $g_a \neq 1$ to ensure it is a generator. In fact, Lemma 1 shows that $x_1+x_3+x_4$ is random over $\mathbb{Z}_q$ even in the face of active attacks. Hence, $g_a \neq 1$ is implicitly guaranteed by the probability. The chance of $g_a = 1$ is extremely minuscule – on the order of $2^{-160}$ for 160-bit $q$. Symmetrically, the same argument applies to the Bob's case. For the same reason, it is implicitly guaranteed by probability that $x_1+x_3 \neq 0$, hence $K = g^{(x_1+x_3)\cdot x_2 \cdot x_4 \cdot s} \neq 1$ holds with an exceedingly overwhelming probability.

**Theorem 2 (Off-line dictionary attack resistance against active attacks).** *Under the Decision Diffie-Hellman (DDH) assumption, provided that $g^{x_1+x_3+x_4}$ is a generator, Bob cannot distinguish Alice's ciphertext $\mathcal{A} = g^{(x_1+x_3+x_4)\cdot x_2 \cdot s}$ from a random non-identity element in the group $G$.*

*Proof.* Suppose Alice is communicating to an attacker (Bob) who does not know the password. The data available to the attacker include $g^{x_1}$, $g^{x_2}$, $\mathcal{A} = g_a^{(x_1+x_3+x_4)\cdot x_2 \cdot s}$ and Zero Knowledge Proofs (ZKP) for the respective exponents. The ZKP only reveals one bit: whether the sender knows the discrete logarithm[5]. Given that $g^{x_1+x_3+x_4}$ is a generator, we have $x_1 + x_3 + x_4 \neq 0$. From Lemma 1, $x_1 + x_3 + x_4$ is a random value over $\mathbb{Z}_q$. So, $x_1 + x_3 + x_4 \in_R [1, q-1]$, unknown to Bob. By protocol definition, $x_2 \in_R [1, q-1]$ and $s \in [1, q-1]$, hence $x_2 \cdot s \in_R [1, q-1]$, unknown to Bob. Based on the Decision Diffie-Hellman assumption [29], Bob cannot distinguish $\mathcal{A}$ from a random non-identity element in the group. □

The above theorem indicates that if Alice is talking directly to an attacker, she does not reveal any useful information about the password. Based on the protocol symmetry, the above results can be easily adapted from Alice's perspective – Alice cannot compute $(x_1 + x_2 + x_3)$, nor distinguish $\mathcal{B}$ from a random element

[5] It should be noted that if we choose Schnorr's signature to realize ZKPs, we implicitly assume a random oracle (i.e., a secure hash function), since Schnorr's signature is provably secure under the random oracle model [30].

in the group. However, the off-line dictionary attack resistance against an active attacker does not necessarily imply resistance against a passive attacker (in the former case, the two passwords are different, while in the latter, they are the same). Therefore, we need the following theorem to show if Alice is talking to authentic Bob, there is no information leakage on the password too.

**Theorem 3 (Off-line dictionary attack resistance against passive attacks).** *Under the DDH assumption, given that $g^{x_1+x_3+x_4}$ and $g^{x_1+x_2+x_3}$ are generators, the ciphertexts $\mathcal{A} = g^{(x_1+x_3+x_4)\cdot x_2\cdot s}$ and $\mathcal{B} = g^{(x_1+x_2+x_3)\cdot x_4\cdot s}$ do not leak any information for password verification.*

*Proof.* Suppose Alice is talking to authentic Bob who knows the password. We need to show a passive attacker cannot learn any password information by correlating the two users' ciphertexts. Theorem 2 states that Bob cannot distinguish $\mathcal{A}$ from a random value in $G$. This implies that even Bob cannot computationally correlate $\mathcal{A}$ to $\mathcal{B}$ (which he can compute). Of course, a passive attacker cannot correlate $\mathcal{A}$ to $\mathcal{B}$. Therefore, to a passive attacker, $\mathcal{A}$ and $\mathcal{B}$ are two random and independent values in $G$; they do not leak any useful information for password verification. □

### 4.2 Forward Secrecy

Next, we discuss the forward secrecy. In the following theorem, we consider a passive attacker who knows the password secret $s$. As we explained earlier, the ZKPs in the protocol require Alice and Bob know the values of $x_1$ and $x_3$ respectively, hence $x_1 + x_3 \neq 0$ (thus $K \neq 1$) holds with an exceedingly overwhelming probability even in the face of active attacks.

**Theorem 4 (Forward secrecy).** *Under the Square Computational Diffie-Hellman (SCDH) assumption[6], given that $K \neq 1$, the past session keys derived from the protocol remain incomputable even when the secret $s$ is later disclosed.*

*Proof.* After knowing $s$, the passive attacker wants to compute $\kappa = H(K)$ given inputs: $\{g^{x_1}, g^{x_2}, g^{x_3}, g^{x_4}, g^{(x_1+x_3+x_4)\cdot x_2}, g^{(x_1+x_2+x_3)\cdot x_4}\}$.

Assume the attacker is able to compute $K = g^{(x_1+x_3)\cdot x_2\cdot x_4}$ from those inputs. For simplicity, let $x_5 = x_1 + x_3 \mod q$. Since $K \neq 1$, we have $x_5 \neq 0$. The attacker behaves like an oracle – given the ordered inputs $\{g^{x_2}, g^{x_4}, g^{x_5}, g^{(x_5+x_4)\cdot x_2}, g^{(x_5+x_2)\cdot x_4}\}$, it returns $g^{x_5\cdot x_2\cdot x_4}$. This oracle can be used to solve the SCDH problem as follows. For $g^x$ where $x \in_R [1, q-1]$, we query the oracle by supplying $\{g^{-x+a}, g^{-x+b}, g^x, g^{b\cdot(-x+a)}, g^{a\cdot(-x+b)}\}$, where $a, b$ are arbitrary values chosen from $\mathbb{Z}_q$, and obtain $f(g^x) = g^{(-x+a)\cdot(-x+b)\cdot x} = g^{x^3-(a+b)\cdot x^2+ab\cdot x}$. In this way, we can also obtain:

$$\begin{aligned} f(g^{x+1}) &= g^{(x+1)^3-(a+b)\cdot(x+1)^2+ab\cdot(x+1)} \\ &= g^{x^3+(3-a-b)\cdot x^2+(3-2a-2b+ab)\cdot x+1-a-b+ab} \end{aligned}$$

[6] The SCDH assumption is provably equivalent to the Computational Diffie-Hellman (CDH) assumption – solving SCDH implies solving CDH, and vice versa [4].

Now we are able to compute $g^{x^2} = \left(f(g^{x+1}) \cdot f(g^x)^{-1} \cdot g^{(-3+2a+2b)\cdot x-1+a+b-ab}\right)^{1/3}$. This, however, contradicts the SCDH assumption [4], which states that one cannot compute $g^{x^2}$ from $g$, $g^x$ where $x \in_R [1, q-1]$. □

### 4.3 Known Session Security

We now consider the impact of a compromised session. If an attacker is powerful enough to compromise a session, we assume he can learn all session-specific secrets, including the raw session key $K$ and ephemeral private keys. In this case, the password will inevitably be disclosed (say by exhaustive search). This is an inherent threat and applies to all the existing PAKE protocols [1, 33, 34, 35, 5, 10, 16, 7, 17, 28].

Still, we shall minimize the impact of a compromised session: in particular, a corrupted session must not harm the security of other established sessions. In the J-PAKE protocol, the raw session key $K = g^{(x_1+x_3)\cdot x_2 \cdot x_4 \cdot s}$ is determined by the ephemeral random inputs $x_1, x_2, x_3, x_4$ from both parties in the session. As we mentioned earlier, the probability has implicitly guaranteed that $K \neq 1$ even in the face of active attacks. The following theorem shows that the obtained session key $K$ is random too – in other words, the session keys are all independent. Therefore, compromising a session (hence learning all session-specific secrets) has no effect on other established session keys.

**Theorem 5 (Random session key).** *Under the Decision Diffie-Hellman (DDH) assumption, given that $K \neq 1$, the past session key derived from the protocol is indistinguishable from a random non-identity element in $G$.*

*Proof.* By protocol definition, $x_2, x_4 \in_R [1, q-1]$, and $s \in [1, q-1]$. Since $K = g^{(x_1+x_3)\cdot x_2 \cdot x_4 \cdot s} \neq 1$, we have $x_1 + x_3 \neq 0$. Let $a = x_1 + x_3$ and $b = x_2 \cdot x_4 \cdot s$. Obviously, $a \in_R [1, q-1]$ and $b \in_R [1, q-1]$. Based on the Decision Diffie-Hellman assumption [29], the value $g^{a\cdot b}$ is indistinguishable from a random non-identity element in the group. □

### 4.4 On-Line Dictionary Attack Resistance

Finally, we study an active attacker, who directly engages in the protocol execution. Without loss of generality, we assume Alice is honest, and Bob is compromised (i.e., an attacker).

In the protocol, Bob demonstrates that he knows $x_4$ and the exponent of $g_b$, where $g_b = g^{x_1+x_2+x_3}$. Therefore, the format of the ciphertext sent by Bob can be described as $\mathcal{B}' = {g_b}^{x_4 \cdot s'}$, where $s'$ is a value that Bob (the attacker) can choose freely.

**Theorem 6 (On-line dictionary attack resistance).** *Under the SCDH assumption, an active attacker cannot compute the session key if he chose a value $s' \neq s$.*

**Table 1.** Summary of J-PAKE security properties

| Modules | Security property | Attacker type | Assumptions |
|---|---|---|---|
| Schnorr signature | leak 1-bit: whether sender knows discrete logarithm | passive/active | DL and random oracle |
| Password encryption | indistinguishable from random | passive/active | DDH |
| Session key | incomputable | passive | CDH |
| | incomputable | passive (know $s$) | CDH |
| | incomputable | passive (know other session keys) | CDH |
| | incomputable | active (if $s' \neq s$) | CDH |
| Key confirmation | leak nothing | passive | – |
| | leak 1-bit: whether $s' = s$ | active | CDH |

*Proof.* After receiving $\mathcal{B}'$, Alice computes

$$K' = (\mathcal{B}'/g^{x_2 \cdot x_4 \cdot s})^{x_2} \tag{1}$$

$$= g^{x_1 \cdot x_2 \cdot x_4 \cdot s'} \cdot g^{x_2 \cdot x_3 \cdot x_4 \cdot s'} \cdot g^{{x_2}^2 \cdot x_4 \cdot (s'-s)} \tag{2}$$

To obtain a contradiction, we reveal $x_1$ and $s$, and assume that the attacker is now able to compute $K'$. The attacker behaves as an oracle: given inputs $\{g^{x_2}, x_1, x_3, x_4, s, s'\}$, it returns $K'$. Note that the oracle does not need to know $x_2$, and it is still able to compute $\mathcal{A} = g^{(x_1+x_3+x_4)\cdot x_2 \cdot s}$ and $\mathcal{B}' = g^{(x_1+x_2+x_3)\cdot x_4 \cdot s'}$ internally. Thus, the oracle can be used to solve the Square Computational Diffie-Hellman problem by computing $g^{{x_2}^2} = (K'/(g^{x_1 \cdot x_2 \cdot x_4 \cdot s'} \cdot g^{x_2 \cdot x_3 \cdot x_4 \cdot s'}))^{{x_4}^{-1}(s'-s)^{-1}}$. Here[7], $x_4 \neq 0$ and $s' - s \neq 0$. This, however, contradicts the SCDH assumption [4], which states that one cannot compute $g^{{x_2}^2}$ from $g$, $g^{x_2}$ where $x_2 \in_R [1, q-1]$. So, even with $x_1$ and $s$ revealed, the attacker is still unable to compute $K'$ (and hence cannot perform key confirmation later). □

The above theorem shows that what an on-line attacker can learn from the protocol is only minimal. Because of the knowledge proofs, the attacker is left with the only freedom to choose an arbitrary $s'$. If $s' \neq s$, he is unable to derive the same session key as Alice. During the later key confirmation process, the attacker will learn one-bit information: whether $s'$ and $s$ are equal. This is the best that any PAKE protocol can possibly achieve, because by nature we cannot stop an imposter from trying a random guess of password. However, consecutively failed guesses can be easily detected, and thwarted accordingly. The security properties of our protocol are summarized in Table 1.

## 5 Comparison

In this section, we compare our protocol with two other balanced PAKE schemes: EKE and SPEKE. These two techniques have several variants, which follow very

[7] This explains why in the protocol definition we need $x_4 \neq 0$, and symmetrically, $x_2 \neq 0$.

**Table 2.** Computational cost for Alice in J-PAKE

| Item | Description | No of Exp |
|---|---|---|
| 1 | Compute $\{g^{x_1}, g^{x_2}\}$ and KPs for $\{x_1, x_2\}$ | 4 |
| 2 | Verify KPs for $\{x_3, x_4\}$ | 4 |
| 3 | Compute $\mathcal{A}$ and KP for $\{x_2 \cdot s\}$ | 2 |
| 4 | Verify KP for $\{x_4 \cdot s\}$ | 2 |
| 5 | Compute $\kappa$ | 2 |
| | Total | **14** |

similar constructs [7]. However, it is beyond the scope of this paper to evaluate them all. Also, we will not compare with augmented schemes (e.g., A-EKE, B-SPEKE, SRP, AMP and OPAKE [27]) due to different design goals.

The EKE and SPEKE are among the simplest and most efficient PAKE schemes. Both protocols can be executed in one round, while J-PAKE requires two rounds. On the computational aspect, both protocol require each user to perform only two exponentiations, compared with 14 exponentiations in J-PAKE (see Table 2).

At first glance, the J-PAKE scheme looks too computationally expensive. However, note that both the EKE and SPEKE protocols must use long exponents (see Section 2.2). Since the cost of exponentiation is linear with the bit-length of the exponent [29], for a typical 1024-bit $p$ and 160-bit $q$ setting, one exponentiation in an EKE or SPEKE is equivalent in cost to 6-7 exponentiations in a J-PAKE. Hence, the overall computational costs for EKE, SPEKE and J-PAKE are actually about the same.

There are several ways to improve the J-PAKE performance. First, the protocol enumerates 14 exponentiations for each user, but actually many of the operations are merely repetitions. To explain this, let the bit length of the exponent be $L = \log_2 q$. Computing $g^{x_1}$ alone requires roughly $1.5L$ multiplications which include $L$ square operations and $0.5L$ multiplications of the square terms. However, the same square operations need not be repeated for other items with the same base $g$ (i.e., $g^{x_2}$ etc). This provides plenty room for efficiency optimization in a practical implementation. In contrast, the same optimization is not applicable to the EKE and SPEKE. Second, it would be more efficient, particularly on mobile devices, to implement J-PAKE using Elliptic Curve Cryptography (ECC). Using ECC essentially replaces the multiplicative cyclic group with an additive cyclic group defined over some elliptic curve. The basic protocol construction remains unchanged.

## 6 Design Considerations

One notable feature of the J-PAKE design is the use of the Zero Knowledge Proof (ZKP), specifically: Schnorr Signature [30]. The ZKP is a well-established cryptographic primitive [9]. For over twenty years, this primitive has been playing a pivotal role in general two/multi-party secure computations [38].

Authenticated key exchange is essentially a two-party secure computation problem. However, the use of ZKP in this area is rare. The main concern is on efficiency: the ZKP is perceived as computationally expensive. So far, almost all of the past PAKE protocols have avoided using ZKP for exactly the reason.

However, the use of ZKP does not necessarily mean the protocol must be inefficient. This largely depends on how to effectively integrate this primitive into the overall design. In our construction, we introduced a novel juggling technique: arranging the random public keys in such a structured way that the randomization factors vanish when both sides supplied the same password. (A similar use of this juggling technique can be traced back to [15] and [8]). As we have shown, this leads to computational efficiency that is comparable to the EKE and SPEKE protocols. To our best knowledge, this design is significantly different from all past PAKE protocols. In the area of PAKE research – which has been troubled by many patent arguments surrounding existing schemes [13] – a new construct may be helpful.

With the same juggling idea, the current construction of the J-PAKE protocol seems close to the optimum. Note in the protocol, we used four $x$ terms – $x_1, x_2, x_3, x_4$. As if one cannot juggle with only two balls, we find it difficult to juggle with two $x$ terms. This is not an issue in the multi-party setting where there are at least three participants (each participant generates one "ball") [15]. For the two-party case, our solution was to let each user create two ephemeral public keys, and thus preserve the protocol symmetry. It seems unlikely that one could improve the protocol efficiency by using a total of only 3 (or even 2) $x$ terms. However, we do not have a proof of minimality on this, so we leave the question open.

## 7 Conclusion

In this paper, we proposed a protocol, called J-PAKE, which authenticates a password with zero-knowledge and then subsequently creates a strong session key if the password is correct. We showed that the protocol fulfills the following properties: it prevents off-line dictionary attacks; provides forward secrecy; insulates a compromised session from affecting other sessions; and strictly limits an active attacker to guess only one password per protocol execution. As compared to the de facto internet standard SSL/TLS, J-PAKE is more lightweight in password authentication with two notable advantages: 1). It requires no PKI deployments; 2). It protects users from leaking passwords (say to a fake bank website).

## Acknowledgments

We thank Ross Anderson and Piotr Zieliński for very helpful comments and discussions.

## References


1. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 191–208. Springer, Heidelberg (2005)
2. Anderson, R.J., Needham, R.: Robustness principles for public key protocols. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 236–247. Springer, Heidelberg (1995)
3. Anderson, R.J.: Security Engineering: A Guide to Building Dependable Distributed Systems. Wiley, New York (2001)
4. Bao, F., Deng, R.H., Zhu, H.: Variations of Diffie-Hellman problem. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003. LNCS, vol. 2836, pp. 301–312. Springer, Heidelberg (2003)
5. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
6. Boneh, D.: The decision Diffie-Hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998)
7. Boyd, C., Mathuria, A.: Protocols for authentication and key establishment. Springer, Heidelberg (2003)
8. Chaum, D.: The dining cryptographers problem: unconditional sender and recipient untraceability. Journal of Cryptology 1(1), 65–67 (1988)
9. Camenisch, J., Stadler, M.: Proof systems for general statements about discrete logarithms, Technical report TR 260, Department of Computer Science, ETH Zürich (March 1997)
10. Bellovin, S., Merritt, M.: Encrypted Key Exchange: password-based protocols secure against dictionary attacks. In: Proceedings of the IEEE Symposium on Research in Security and Privacy (May 1992)
11. Bellovin, S., Merritt, M.: Augmented Encrypted Key Exchange: a password-based protocol secure against dictionary attacks and password file compromise. In: Proceedings of the 1st ACM Conference on Computer and Communications Security, pp. 244–250 (November 1993)
12. Bellovin, S., Merritt, M.: Cryptographic protocol for secure communications, U.S. Patent 5,241,599
13. Ehulund, E.: Secure on-line configuration for SIP UAs, Master thesis, The Royal Institute of Technology (August 2006)
14. Ford, W., Kaliski, B.S.: Server-assisted generation of a strong secret from a password. In: Proceedings of the 9th International Workshops on Enabling Technologies, pp. 176–180. IEEE Press, Los Alamitos (2000)
15. Hao, F., Zieliński, P.: A 2-round anonymous veto protocol. In: Proceedings of the 14th International Workshop on Security Protocols, SPW 2006, Cambridge, UK (May 2006)
16. Jablon, D.: Strong password-only authenticated key exchange. ACM Computer Communications Review 26(5), 5–26 (1996)
17. Jablon, D.: Extended password protocols immune to dictionary attack. In: Proceedings of the WETICE 1997 Enterprise Security Workshop, pp. 248–255 (June 1997)
18. Jablon, D.: Cryptographic methods for remote authentication, U.S. Patent 6,226,383 (March 1997)
19. Jablon, D.: Password authentication using multiple servers. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 344–360. Springer, Heidelberg (2001)

20. Jaspan, B.: Dual-workfactor Encrypted Key Exchange: efficiently preventing password chaining and dictionary attacks. In: Proceedings of the Sixth Annual USENIX Security Conference, pp. 43–50 (July 1996)
21. Kobara, K., Imai, H.: Pretty-simple password-authenticated key-exchange under standard assumptions. IEICE Transactions E85-A(10), 2229–2237 (2002)
22. Van Oorschot, P.C., Wiener, M.J.: On Diffie-Hellman key agreement with short exponents. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 332–343. Springer, Heidelberg (1996)
23. Patel, S.: Number theoretic attacks on secure password schemes. In: Proceedings of the IEEE Symposium on Security and Privacy (May 1997)
24. Perlman, R., Kaufman, C.: Secure password-based protocol for downloading a private key. In: Proceedings of the Network and Distributed System Security (February 1999)
25. MacKenzie, P.: The PAK suite: protocols for password-authenticated key exchange, Technical Report 2002-46, DIMACS (2002)
26. MacKenzie, P.: On the Security of the SPEKE Password-Authenticated Key Exchange Protocol. Cryptology ePrint Archive: Report 057 (2001)
27. IEEE P1363 Working Group, P1363.2: Standard Specifications for Password-Based Public-Key Cryptographic Techniques. Draft available at, `http://grouper.ieee.org/groups/1363/`
28. Wu, T.: The Secure Remote Password protocol. In: Proceedings of the Internet Society Network and Distributed System Security Symposium, pp. 97–111 (March 1998)
29. Stinson, D.: Cryptography: theory and practice, 3rd edn. Chapman & Hall/CRC (2006)
30. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
31. Zhang, M.: Analysis of the SPEKE password-authenticated key exchange protocol. IEEE Communications Letters 8(1), 63–65 (2004)
32. 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)
33. Goldreich, O., Lindell, Y.: Session-key generation using human passwords only. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 408–432. Springer, Heidelberg (2001)
34. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 475–494. Springer, Heidelberg (2001)
35. Jiang, S.Q., Gong, G.: Password based key exchange with mutual authentication. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 267–279. Springer, Heidelberg (2004)
36. Krawczyk, H.: HMQV: a high-performance secure Diffe-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
37. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton (1996)
38. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Proceedings of the Nineteenth Annual ACM Conference on Theory of Computing, pp. 218–229 (1987)
39. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003)