The SPEKE Protocol Revisited

Feng Hao, Siamak F. Shahandashti

Newcastle University, UK
{feng.hao, siamak.shahandashti}@ncl.ac.uk

Abstract. The SPEKE protocol is commonly considered one of the classic Password Authenticated Key Exchange (PAKE) schemes. It has been included in international standards (particularly, ISO/IEC 11770-4 and IEEE 1363.2) and has been deployed in commercial products. We observe that the original SPEKE specification is subtly different from those defined in the ISO/IEC 11770-4 and IEEE 1363.2 standards. We show that those differences have critical security implications. First of all, we present two new attacks on SPEKE: a relay attack and a key-malleability attack. The first attack allows an attacker to impersonate a user without knowing the password by engaging in two parallel sessions with the victim. The second attack allows an attacker to malleate the session key established between two honest users without being detected. Both attacks are applicable to the original SPEKE scheme. However, they are to some extent addressed in the ISO/IEC 11770-4 and IEEE 1363.2 standards, but in a vaguely defined manner. The vagueness makes it extremely difficult for a security-conscious developer to implement the protocol correctly. We propose countermeasures and suggest concrete changes to the standards.

1 Introduction

Password Authenticated Key Exchange (PAKE) is a protocol that aims to establish a secure communication channel between two remote parties based on a shared low-entropy password without relying on any external trusted parties. Since the seminal work by Belloven and Merrit in 1992 [2], many PAKE protocols have been proposed, and some have been standardised [10, 11].

The Simple Password Exponential Key Exchange (SPEKE) protocol is one of the most well-known PAKE solutions. It was originally designed by Jablon in 1996 [7]. Although some concerns are raised [8, 9], no major flaws seems to have been uncovered. Over the past decade, SPEKE has been included in the IEEE P1362.2 [10] standard draft[1] and ISO/IEC 11770-4:2006 [11] Part 4: "Mechanisms based on weak secrets". Furthermore, SPEKE has been deployed in commercial applications – for example in BlackBerry devices produced by Research in Motion and in Entrust's TruePass [5] end-to-end web products [6].

[1] As of July 2014, the latest draft is D26. See `http://grouper.ieee.org/groups/1363/passwdPK/draft.html`

In this paper, we revisit the original SPEKE protocol and review its specifications in the two standardisation documents: IEEE P1363.2 and ISO/IEC 11770-4. We observe that the original protocol is subtly different from those defined in the standards. The reason for the difference, or deviation from the original specification, is not justified clearly in the standards.

During the investigation, we have identified several issues with SPEKE that have not been reported before. Our findings are summarised below:

1. We show the original SPEKE protocol is subject to a relay attack when the victim is engaged in two parallel sessions with an active attacker. The attack is able to achieve mutual authentication in both sessions without knowing the password.
2. We show the original SPEKE protocol is subject to a key-malleability attack. The attacker, sitting in between two honest users, is able to malleate the session key without being detected.
3. While both attacks clearly work on the original SPEKE protocol, we show they are to some extent addressed in IEEE P1363.2 and ISO/IEC 11770-4, but in a vaguely defined way. We propose explicit and concrete changes to both standards.

Details of our findings are explained in the following sections.

2 The original SPEKE scheme

First, we define the original SPEKE scheme based on Jablon's 1996 paper [7]. Let p be a safe prime, $p = 2q+1$ where q is also a prime. Assume two remote parties, Alice and Bob, share a common password s. SPEKE defines a function $f(\cdot)$ to map a password s to a group element: $f(s) = s^2 \bmod p$. We use g to denote the result returned from $f(s)$, i.e., $g = f(s)$. The SPEKE protocol provides implicit authentication in one round, which is defined below. (Unless stated otherwise, all modular operations are performed modulo p, hence the explicit $\bmod p$ is omitted for simplicity.)

Round 1 (SPEKE) *Alice selects* $x \in_R [1, q-1]$ *and sends* g^x *to Bob. Similarly, Bob selects* $y \in_R [1, q-1]$ *and sends* g^y *to Alice.*

Upon receiving the sent data, Alice verifies that g^y is within $[2, p-2]$. This is to ensure the received element does not fall into the small subgroup of order two, which contains $\{1, p-1\}$. Alice then computes a session key $\kappa = H((g^y)^x) = H(g^{xy})$, where H is a secure one-way hash function. Similarly, Bob verifies that g^x is within $[2, p-2]$. He then computes the same session key $\kappa = H((g^x)^y) = H(g^{xy})$.

To provide explicit key confirmation, the SPEKE paper defines the following procedure. One party sends $H(H(\kappa))$ and the other party replies with $H(\kappa)$. The paper does not specify who must initiate the key confirmation and hence leaves it as a free choice for specific applications to decide.

3 Previously reported attacks

In 2004, eight years after SPEKE was initially designed, Zhang presented an exponential-equivalence attack [8]. The attack is based on the observation that some passwords are exponentially equivalent. Hence, an active attacker can exploit that equivalence to test multiple passwords in one protocol execution. This is especially problematic when the password is digits-only, e.g., a Personal Identification Numbers (PIN). As a countermeasure, Zhang proposed to hash the password before taking the square operation. In other words, redefine the password mapping function as: $f(s) = (H(s))^2 \bmod p$. The hashing of passwords makes it much harder for the attacker to find exponential equivalence among the hashed outputs. Zhang's attack is acknowledged in IEEE P1363.2 [10], which adds a hash function in SPEKE when deriving the base generator from the password.

In 2005, Tang and Mitchell presented three attacks on SPEKE [9]. The first attack is similar to Zhang's [8] – an on-line attacker tests multiple passwords in one execution of the protocol by exploiting the exponential equivalence of some passwords. The second attack assumes that the user shares the same password with two servers, say $S1$ and $S2$. By relaying the messages between the client and $S2$, the attacker may trick the client into believing that she shares a key with $S1$, but actually the key is shared with $S2$. The authors call this an "unknown key-share" attack. They suggest to address this attack by including the server's identifier into the computation of g. (However, we note that this suggested countermeasure has the side-effect of breaking the symmetry of the original protocol.) The third attack indicates a generic vulnerability. In this scenario, two honest parties launch two concurrent sessions. The attacker can swap the messages between the two sessions to exchange the two session keys. The two communicating parties will be able to decrypt messages successfully but they may get confused about which message belongs to which session.

4 New attacks

In this section, we describe two new attacks: relay attack and key-malleability attack. The first attack indicates a significant flaw in the original design of SPEKE, while the second attack has an unfavourable implication on the theoretical analysis of the protocol.

4.1 Relay attack

The Relay attack works when the user is engaged in several sessions in parallel with another user. This is a realistic scenario in practice as two users may want to run several concurrent SPEKE key exchange sessions and use each established channel for a specific application, as Tang and Mitchell also observe [9].

Without loss of generality, we assume Alice is honest and Bob is an attacker. Bob does not know the password but attempts to impersonate someone who

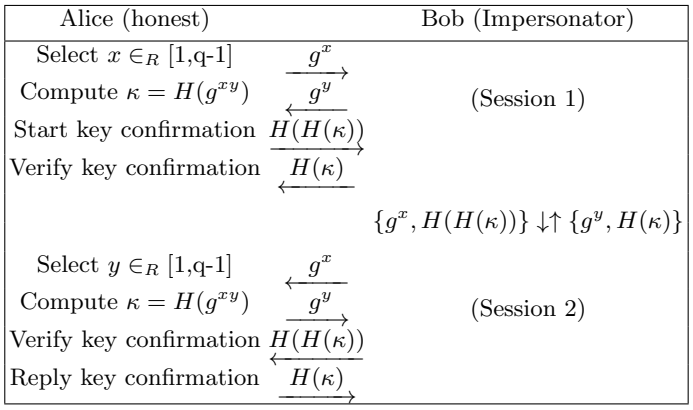

Fig. 1. Relay attack on SPEKE

knows the password. We let Alice initiate a SPEKE session – which we call Session 1 – with Bob by sending g^x (see Figure 1). At the same time, Bob initiates another SPEKE session – which we call Session 2 – with Alice by relaying g^x. In the second session, Alice replies with g^y, which is then relayed by Bob as a Bob's message in Session 1. Following the key confirmation procedure as in the original SPEKE paper, Alice provides the first key confirmation challenge in Session 1 $H(H(\kappa))$, which is subsequently relayed to Section 2 as Bob's key confirmation challenge. In Session 2, Alice answers the key confirmation challenge by replying with $H(\kappa)$, which is then relayed in Section 1 to complete the mutual authentication in both sessions. Without knowing the password, Bob has been successfully authenticated by Alice. This indicates a significant flaw in the original protocol specification.

The relay attack in Fig. 1 can be made slightly more complex. When relaying g^x received from Section 1 to Alice in Section 2, the attacker can raise it to the power of an integer z, so it becomes g^{xz}. Accordingly, when relaying g^y received Section 2 to Alice in Section 1, the attacker can raise it to the power of the same integer z, so it becomes g^{yz}. The attack works the same as before except that the key exchange messages in two sessions are different. Hence, simply checking duplicates of data may not be an effective countermeasure.

The relay attack is similar to the "unknown key-share" attack in Tang-Mitchell's paper [9], however, our attack seems to be more feasible and harmful than theirs. The main difference is that in our attack, the attacker relays the user's message back to the user herself. In essence, this relay-to-self attack is basically the same as the "wormhole attack" [3], in which the attacker relays the sender's message back to the sender to pass authentication. However, the "wormhole attack" presented in [3] works in a PKI-based key exchange setting while the relay attack reported here happens in a password-based key exchange setting. The two settings are distinct.

Alice		MITM		Bob
Select $x \in_R$ [1,q-1]	$\xrightarrow{g^x}$	Select $z \in [2, q-2]$		
		Raise to power z	$\xrightarrow{(g^x)^z}$	Check $(g^x)^z \in [2, q-2]$
				Select $y \in_R [1, q-1]$
Check $(g^y)^z \in [2, q-2]$	$\xleftarrow{(g^y)^z}$	Raise to power z	$\xleftarrow{g^y}$	
Compute $\kappa = H(g^{xyz})$				Compute $\kappa = H(g^{xyz})$

Fig. 2. Key-malleability attack on SPEKE

4.2 Key-malleability attack

A second attack is called the key-malleability attack. In this attack, the attacker sits in the middle between two honest users (see Figure 2). The attacker chooses an arbitrary z within the range of $[2, q-1]$, raises the intercepted item to the power of z and passes it on. The users at two ends are still able to derive the same session key $\kappa = H(g^{xyz})$, but without being aware that the messages have been modified.

We do not claim there is a direct practical harm caused by this attack. However, the fact that an attacker is able to malleate the session key without being detected may have significant implications on the theoretical analysis of the protocol. In the original SPEKE paper, the protocol comes with no security proofs. However, it is heuristically argued that the security of the session key in SPEKE depends on either the Computational Diffie-Hellman assumption (i.e., an attacker is unable to compute the session key) or the Decisional Diffie-Hellman assumption (i.e., an attacker is unable to distinguish the session key from random). The existence of such a key-malleability attack suggests that a tight reduction to CDH or DDH may be impossible (currently, formal reductionist proofs for SPEKE do not exist). The attacker's ability to inject randomness into the session key without being noticed can significantly complicate the theoretical analysis. As an example, let us assume the attacker chooses z as a result of a function with input g^x, i.e., $z = f(g^x)$. Because of the correlation between x and z on the exponent, standard CDH and DDH are no longer applicable here.

5 Discussion

While the two attacks clearly work on the original SPEKE protocol [7], it may be argued that they do not work on variants of SPEKE defined in IEEE P1363.2 and ISO/IEC 11770-4. In this section, we explain the difference between the original protocol and its variants in the standards in relation to the two attacks.

5.1 Explicit key confirmation

First of all, we observe that the key confirmation procedure of SPEKE defined in the standard is different from that in the original SPEKE paper [7]. For example,

in ISO/IEC 11770-4, the key confirmation works as follows [11] (the procedure in IEEE P1363.2 [10] is basically the same).

$$\begin{aligned} \text{Alice} \rightarrow \text{ Bob } &: H(\text{“}0x03\text{”} \| g^x \| g^y \| g^{xy} \| g) \\ \text{Bob } \rightarrow \text{Alice} &: H(\text{“}0x04\text{”} \| g^x \| g^y \| g^{xy} \| g) \end{aligned}$$

As explicitly stated in the ISO/IEC 11770-4 standard, there is no order in the above two steps[2]. Either party is free to send out the key confirmation message without waiting for the other party.

Effect on relay attack We observe that the above key confirmation procedure does not prevent the relay attack. The attacker is still able to relay the key confirmation string in one session to another parallel session to accomplish mutual authentication in both sessions without being detected. The attack works largely because the session keys are identical in the two sessions.

Effect on Key-malleability attack The key-malleability attack no longer works with the key confirmation procedure defined in ISO/IEC 11770-4 (and IEEE P1363.2). However, it is worth noting that the key confirmation procedures in both standards are marked as “optional”. Hence, the key-malleability is not completely addressed.

5.2 Definition of Password

In the original SPEKE paper, the mapping of a password s to a group element over the prime field is simply achieved by $f(s) = s^2$. To prevent Zhang's exponential-equivalence attack, it is necessary to add a hash function before performing the squaring operation, i.e., $f(s) = (H(s))^2$. This is essentially the mapping function defined in ISO/IEC 11770-4 and IEEE P1363.2 (for the case that p is a safe prime). However, the definition of the shared secret s is subtly changed in both standards. For example, in ISO/IEC 11770-4, the shared low-entropy secret is defined as follows:

> *“A password-based octet string which is generally derived from a password or a hashed password, identifiers for one or more entities, an identifier of a communication session if more than one session might execute concurrently, and optionally includes a salt value and/or other data.”*[3]

The definition of the shared low-entropy secret (denoted π in the standard document) in IEEE P1363.2 is similar:

---

[2] In the same standard, it is also stated that there is no order during the SPEKE exchange phrase. We find the two statements contradictory: the fact that g^x comes before g^y in the definition of key confirmation implies there is an order during the key exchange phase.

[3] It is explained in a note in ISO/IEC 11770-4 that the inclusion of the identifiers is to avoid an unknown key-share attack reported in Tang-Mitchell's paper [9].

> *"A password-based octet string, used for authentication. π is generally derived from a password or a hashed password, and may incorporate a salt value, identifiers for one or more parties, and/or other shared data."*

Effect on relay attack Strictly speaking, if the entity identifiers and the session identifier are included in the definition of the shared secret s, the relay attack presented in Section 4 will not work. While the lawlike wording in both standards looks rigorous, it does not really help the implementer, as the instruction is extremely vague. This is unsatisfactory, especially because the detail here has critical security implications. To start with, it is not even clear if one or both parties' identifiers should be included, and if only one identifier needs to be included, which one.

For a more concrete discussion, let us denote Alice's identifier as $\hat{A}$, Bob's identifier as $\hat{B}$ and the session identifier as SID. One straightforward way to include all these identifiers is: $s = H(\text{Password}\|\hat{A}\|\hat{B}\|\text{SID})$. But this implies a preferred order of the parties' identifies, which need to be agreed beforehand. A slightly better definition is as follow: $s = H(\text{Password}\|\min(\hat{A},\hat{B})\|\max(\hat{A},\hat{B})\|\text{SID})$. Yet it remains questionable how the SID should be defined and by whom. If SID should be included in the password-based string, this seems to contradict the definition in the standards that the password-based string is part of the "prior shared parameters" before the key exchange. In conclusion, we consider the SPEKE protocol definition in both standards as "under-specified".

Effect on Key-malleability attack The inclusion of identifiers for one or more entities and the specific session into the definition of the password-based string has no effect in preventing the key-malleability attack. The attack still works.

5.3 Countermeasures and suggested changes in standards

First of all, we need to understand the causes of the two attacks. There are several reasons. First, there is no reliable method in SPEKE to prevent a sent message being relayed back to the sender. Second, there is no mechanism in the protocol to verify the integrity of the message, i.e., whether they have been altered during the transit. Third, no user identifiers are included in the key exchange process. It may be argued that all these issues can be addressed by using Zero Knowledge Proof (ZKP) (as done in [4]). However, in SPEKE, the generator is a secret, which makes it incompatible with any existing ZKP construction. Since the use of ZKP is impossible in SPEKE, we need to address the attacks in a different way.

Our proposed solution is to redefine the session key computation. Assume Alice sends $M = g^x$ and Bob sends $N = g^y$. The session key computation is defined as follows:

$$\kappa = H\left(\min(\hat{A},\hat{B}), \max(\hat{A},\hat{B}), \min(M,N), \max(M,N), g^{xy}\right) \tag{1}$$

The patched SPEKE protocol is summarized in Fig. 3. When the two users are engaged in multiple concurrent sessions, they need to ensure the identifiers

Alice ($\hat{A}$)		Bob ($\hat{B}$)
Select $x \in_R$ [1,q-1]. Compute $M = g^x$	$\hat{A}, M = g^x \xrightarrow{\quad}$	Check $M \in [2, q-2]$
Check $N \in [2, q-2]$	$\xleftarrow{\quad} \hat{B}, N = g^y$	Select $y \in$ [1,q-1]. Compute $N = g^y$
Alice Computes: $\kappa_a = H\left(\min(\hat{A},\hat{B}), \max(\hat{A},\hat{B}), \min(M,N), \max(M,N), N^x\right)$		
Bob Computes: $\kappa_b = H\left(\min(\hat{A},\hat{B}), \max(\hat{A},\hat{B}), \min(M,N), \max(M,N), M^y\right)$		

Fig. 3. Patched SPEKE

are unique between these sessions. As an example, assume Alice and Bob launch several concurrent sessions. They may use "Alice" and "Bob" in the first session. When launching a second concurrent session, they should add an extension to make the identifier unique – for example, they may agree at the protocol level to start the extension from "1" and increment by one if a new concurrent session is created. Thus, the actual user identifiers become "Alice-1" and "Bob-1" in the second session. In the third session, the user identifiers become "Alice-2" and "Bob-2", and so on. As long the user identifiers are unique between concurrent sessions, the use of extra session identifier does not seem needed.

The new definition of the session-key computation function in Eq. 1 should address the relay and key-malleability attacks in Section 4 (and also the "unknown-key share" attack and the generic attack reported by Tang and Mitchell [9]). This is achieved without having to involve explicit key confirmation, so the key confirmation can remain as "optional" as it is in the current standards.

There is an alternative solution, which is to make the definition of a shared low-entropy secret more explicit in the standards. One way is to define the shared secret as below:

$$s = H\left(\text{Password} \| \min(\hat{A},\hat{B}) \| \max(\hat{A},\hat{B})\right) \tag{2}$$

In the above definition, we do not include the session identifier SID, as the concept seems to have been absorbed in the user identifiers as long as we ensure the use identifiers are unique between concurrent sessions.

Comparing the two solutions, we prefer the first solution in Eq. 1 (also see Fig. 3) for the following reasons.

- The first solution is more flexible to accommodate pre-computation of g^x and g^y. In the second solution, the user must know the identifier of the other party before the key exchange, which is not always the case.
- The first solution may be more round-efficient. Alice and Bob do not have to know the exact identifier of the other party before starting the key exchange. But in the second solution, Alice and Bob may need an extra round before they are able to compute the generator g.
- The first solution may be more computationally efficient. Because the generator g is unchanged for the same password, it only needs to be computed

once. In comparison, the generator needs to be re-computed with any change in the user identifier. (This may not make much difference in terms of computation if a safe prime is used, but it may significantly decrease performance in other group settings, e.g., in Elliptic Curve.)

A further suggestion we would like to make for both standards is to reconsider the definition of the key confirmation method. The existing method, as defined in ISO/IEC 11770-4 and IEEE 1363.2, breaks the symmetry of the protocol. The key confirmation method in the original SPEKE paper [7] has the same problem.

Our rationale for suggesting a change is not based on the security consideration, but on the ground of system reliability. For example, if the two parties happen to send the first key confirmation message at the same time, i.e., Alice sends $H(H(\kappa))$ and without receiving Alice's message, Bob also sends $H(H(\kappa))$. In that case, they may enter a deadlock and may have to abort the session and restart a new one. The chance of such occurrence may be non-negligible if the latency in the network communication is high.

The solution we propose is based on the key confirmation defined in NIST SP 800-56A Revision 1 [1]. It works as follow:

$$\begin{aligned} \text{Alice} \rightarrow \text{ Bob } &: \text{HMAC}(\kappa, \text{“KC\_1\_U”} \| \hat{A} \| \hat{B} \| g^x \| g^y) \\ \text{Bob } \rightarrow \text{Alice} &: \text{HMAC}(\kappa, \text{“KC\_1\_U”}, \| \hat{B} \| \hat{A} \| g^y \| g^x) \end{aligned}$$

In the above key confirmation method, HMAC is a hash-based MAC algorithm and the string “KC_1_U” refers to unilateral key confirmation [1]. There is no dependence between the two flows, so Alice and Bob can do this in one round.

6 Conclusion

In this paper, we present two new attacks on SPEKE, a protocol that has been included in the IEEE P1363.2 and ISO/IEC 11770-4 standards, and deployed in commercial products. The first attack indicates a significant practical flaw that needs to be addressed, while the second attack has only theoretical implications. We explain the difference between the original SPEKE protocol and its variants defined in both standards and show how the difference is critically relevant to the presented attacks. We suggest concrete changes to both standards to address the issues identified in this paper.

References

1. E. Barker, D. Johnson, and M. Smid, “Recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised)”, NIST Special Publication 800-56A, March 2007. Available at `http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf`

2. S. Bellovin and M. Merritt, "Encrypted Key Exchange: password-based protocols secure against dictionary attacks," Proceedings of the IEEE Symposium on Research in Security and Privacy, May 1992.
3. F. Hao, "On robust key agreement based on public key authentication", Proceedings of the 14th International Conference on Financial Cryptography and Data Security (FC'10), Tenerife, Spain, LNCS 6052, pp. 383-390, 2010.
4. F. Hao, P. Ryan, "Password authenticated key exchange by juggling," Proceedings of the 16th Workshop on Security Protocols (SPW'08), Cambridge, UK, LNCS 6615, pp. 159-171, 2008.
5. "Entrust TruePass Product Portfolio: Strong Authentication, Digital Signatures and end-to-end encryption for the Web Portal," Technical Overview, Entrust Inc., July 2003. Available online at `http://www.entrust.com/wp-content/uploads/2013/05/entrust_truepass_tech_overview.pdf`
6. "BlackBerry Bridge App and BlackBerry PlayBook Tablet," Security Technical Overview - Version 2.0, Research In Motion Ltd., February 2012. Available online through Blackberry Knowledge Base at `http://btsc.webapps.blackberry.com/btsc/microsites/searchEntry.do`
7. D. Jablon, "Strong password-only authenticated key exchange," *ACM Computer Communications Review*, Vol. 26, No. 5, pp. 5-26, October 1996.
8. M. Zhang, "Analysis of the SPEKE password-authenticated key exchange protocol," *IEEE Communications Letters*, Vol. 8, No. 1, pp. 63-65, January 2004.
9. Q. Tang, C. Mitchell, "On the security of some password-based key agreement schemes," International conference on Computational Intelligence and Security (CIS), LNCS Vol. 3802, pp. 149-154, 2005.
10. IEEE P1363 Working Group, P1363.2: Standard Specifications for Password-Based Public-Key Cryptographic Techniques. Draft available at: `http://grouper.ieee.org/groups/1363/`
11. International Standard on Information Technology, Security Techniques, Key Management, Part 4: "Mechanisms based on week secrets," ISO/IEC 11770-4:2006.