*Symmetry* **2015**, *7*, 1587-1594; doi:10.3390/sym7031587

OPEN ACCESS

*symmetry*

ISSN 2073-8994

www.mdpi.com/journal/symmetry

*Article*

# Prevention of Exponential Equivalence in Simple Password Exponential Key Exchange (SPEKE)

**Hanwook Lee [1,2] and Dongho Won [1,*]**

[1] Department of Computer Engineering, Sungkyunkwan University, 2066 Seoburo, Suwon, Gyeonggido 440-746, Koera; E-Mail: hwlee@security.re.kr

[2] Financial Telecommunications & Clearings Research Center, Korea Financial Telecommunications & Clearings Institute, 9, 213 beongil, Jeongjail-ro, Seongnam, Gyeonggido 463-811, Korea

* Author to whom correspondence should be addressed; E-Mail: dhwon@security.re.kr; Tel.: +82-31-290-7213; Fax: +82-31-290-5695.

*Received: 16 June 2015 / Accepted: 28 August 2015 / Published: 2 September 2015*

**Abstract:** Simple Password Exponential Key Exchange (SPEKE) and Dragonfly are simple password-based authenticated key exchange protocols that use a value derived from a shared password as a generator for modular exponentiation, as opposed to Diffie–Hellman key exchange, which uses a fixed value. However, it has been shown that in SPEKE, an active attacker, can examine multiple passwords in a single attempt because the passwords have an exponential correlation. We show that Dragonfly can also suffer from the same problem, and we propose a simple countermeasure to prevent the exponential equivalence in SPEKE.

**Keywords:** Exponential equivalence; password authentication; key exchange; SPEKE; Dragonfly

## 1. Introduction

Password-Authenticated Key Exchange (PAKE) is an authenticated key exchange protocol between two or more parties that share only a human-memorable password without depending on any third party and communicating only via an insecure network. Because the password dictionary is sufficiently small, PAKE protocols are designed with the assumption that an attacker can enumerate all the possible passwords off-line within a reasonable time. The intention of PAKE protocols is to prevent off-line guessing attack. They are considered to be safe when an exhaustive search is the only available means of attack.

Simple Password Exponential Key Exchange (SPEKE), proposed by Jablon [1], and Dragonfly, proposed by Harkins [2], are well-known PAKE protocols. These protocols are similar to Diffie–Hellman key exchange except that they use a value derived from a shared password as a generator for exponentiation, instead of a fixed value. Such a characteristic makes these protocols seem simpler yet more efficient than other PAKE protocols. However, in 2004, Zhang showed that an active attacker can try multiple passwords in a single attempt by exploiting the exponential equivalence of the passwords in SPEKE [3]. Furthermore, in 2005, Tang and Mitchell identified other security vulnerabilities such as unknown key-share attack and generic attack as well as the exponential equivalence problem [4]. Subsequently, in 2014, Hao and Shahandashti showed that SPEKE can suffer from impersonation attack and key-malleability attack [5] and Clark and Hao showed that Dragonfly can suffer from a small subgroup attack [6].

The above-mentioned vulnerabilities except the exponential equivalence problem can be addressed in SPEKE by modifying the key confirmation process [4,5] and in Dragonfly by checking that the received element is a member of the supergroup [6]. A possible solution to the exponential equivalence problem is hashing passwords, as in the case of the revised version of SPEKE, which makes it difficult for an attacker to guess the password [7]. However, this approach does not overcome the essential problem, because the exponential equivalence of passwords persists even after the passwords are hashed. Hao and Ryan claimed that there is no guarantee that an attacker is unable to formulate exponential relationships among hashed passwords; no existing hash function is designed for that purpose [8].

In this paper, we show that Dragonfly can also suffer from the exponential equivalence problem, and we propose a simple solution that can prevent password-guessing attacks based on the exponential equivalence of passwords in the SPEKE protocol.

## 2. Review of SPEKE and Dragonfly

### *2.1. The SPEKE Protocol*

Let $p$ be a safe prime, $p = 2q + 1$, where $q$ is also a prime. SPEKE is defined over a subgroup of $\mathbb{Z}_p^*$ of prime order $q$. Assume that two parties, Alice and Bob, share a common password $\pi$ and want to exchange a password-authenticated key. SPEKE defines a function $f$ to map the password $\pi$ to a generator, *i.e.*, $f(\pi) = \pi^2 \bmod p$ in the original SPEKE [1] or $f(\pi) = (h(\pi))^2 \bmod p$ in the revised version of SPEKE [7], where $h$ is a cryptographic hash function (e.g., SHA-256). Let $g$ denote the generator corresponding to $\pi$, *i.e.*, $g = f(\pi)$. The squaring of $\pi$ or $h(\pi)$ ensures that the generator $g$ does not fall into the small subgroup of order two, which contains $\{1, p-1\}$.

SPEKE is exactly the same as Diffie–Hellman key exchange except that it uses a value derived from $\pi$ as a generator for exponentiation. The details of fully constrained SPEKE are as follows:

1. Alice $\rightarrow$ Bob: Send $A = g^a \bmod p$, where $a$ is randomly selected in $\mathbb{Z}_q^*$.
2. Alice $\leftarrow$ Bob: Send $B = g^b \bmod p$, where $b$ is randomly selected in $\mathbb{Z}_q^*$.
3. Alice & Bob: Compute $K = B^a \bmod p$ and $K = A^b \bmod p$, respectively.

Now, Alice and Bob share the secret $K = g^{ab} \bmod p$.

Alice and Bob confirm that they are sharing the same secret $K$. For this purpose, Alice and Bob exchange validation values $V_A$, $V_B$, and validate them. A hash function is used to generate the values. The details are as follows:

4. Alice $\leftarrow$ Bob: Send $V_B = h(h(h(K)))$.
5. Alice: Verify $V_B$.
6. Alice $\rightarrow$ Bob: Send $V_A = h(h(K))$.
7. Bob: Verify $V_A$.

### *2.2. The Dragonfly Protocol*

Although Dragonfly, based on discrete logarithm cryptography, can be implemented either on a finite field or on an elliptic curve, in this paper we deal with the protocol operated on a finite field.

Let $p$ be a large prime of the form $p = rq + 1$, where $r$ and $q$ relatively prime. Dragonfly is defined over a finite cyclic group $Q$, which is a subgroup of $\mathbb{Z}_p^*$ of prime order $q$. Assume that two parties, Alice and Bob, share a common password $\pi$ and want to exchange a password-authenticated key. Dragonfly defines two functions $f$ and $h_1$ to map the identities of the two participants and the shared password $\pi$ to a generator, *i.e.*, $f(\text{Alice}, \text{Bob}, \pi) = (h_1(\text{Alice}, \text{Bob}, \pi))^r \bmod p$. The functions can be briefly represented as $f : \{0,1\}^* \rightarrow Q$ and $h_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, respectively. (For more details, see [2].) Let $g$ denote the generator, *i.e.*, $g = f(\text{Alice}, \text{Bob}, \pi)$.

The Dragonfly protocol works as follows:

1. Alice$\rightarrow$Bob: Send $A = g^{-a_1} \bmod p$ and $a = (a_1 + a_2) \bmod q$, where $a_1$ and $a_2$ are randomly selected in $\mathbb{Z}_q^*$.

   Alice$\leftarrow$Bob: Send $B = g^{-b_1} \bmod p$ and $b = (b_1 + b_2) \bmod q$, where $b_1$ and $b_2$ are randomly selected in $\mathbb{Z}_q^*$.
2. Alice & Bob: Compute $K = (g^b \cdot B)^{a_2} \bmod p$ and $K = (g^a \cdot A)^{b_2} \bmod p$, respectively.

Now, Alice and Bob share the secret $K = g^{a_2 b_2} \bmod p$.

3. Alice $\rightarrow$ Bob: Send $V_A = h(K||A||a||B||b)$.

   Alice $\leftarrow$ Bob: Send $V_B = h(K||B||b||A||a)$.
4. Alice & Bob: Verify $V_A$ and $V_B$, respectively.

## 3. Exponential Equivalence in SPEKE and Dragonfly

### *3.1. Exponential Equivalence in SPEKE*

In this subsection, we review the problem identified by Zhang, *i.e.*, a single on-line attempt can scan through all the exponentially equivalent passwords [3]. First, if $p$ is a large safe prime, then for arbitrary given $b$, it is extremely difficult to find an integer $x$ that satisfies the following equation:

$$b = a^x \bmod p.$$

However, if mod $p$ is not present, this equation can be converted into a simple equation that yields $\log_a b$. As such, we can say that two elements, $b_1 = a^{x_1}$ and $b_2 = a^{x_2}$, where $a, b_1, b_2 \in Z_p^*$, are exponentially equivalent, and the set of exponentially equivalent elements is identified as the exponential equivalence class. For example, among the numbers less than 100, the exponential equivalence classes are as follows:

$$\{2^1, 2^2, 2^3, 2^4, 2^5, 2^6\}, \{3^1, 3^2, 3^3, 3^4\}, \{5^1, 5^2\}, \{7^1, 7^2\}.$$

Now, let us consider how every element can be examined at once in the SPEKE protocol. Assume that the outputs of the mapping function $f$ for $j$ passwords from $\pi_1$ to $\pi_j$ are exponentially equivalent with a base $m$. The actual class in SPEKE is

$$\{f(\pi_1) = m^{e_1}, f(\pi_2) = m^{e_2}, \cdots, f(\pi_j) = m^{e_j}\},$$

where the exponent $e_i$ for each $i \in \{1, 2, ..., j\}$, is an integer greater than or equal to 1. An active attacker, Eve, masquerading as Alice, generates a random $a'$ and sends $A'$ to Bob when $A' = m^{a'} \bmod p$ is calculated. Bob generates a random $b$ and sends $B$ and $V_B'$ to Eve when $B = (f(\pi))^b \bmod p$, $K' = (A')^b \bmod p$, and $V_B' = h(h(h(K')))$ are calculated. On receiving $B$ and $V_B'$ from Bob, Eve terminates the protocol and for each $i$, $1 \leq i \leq j$, computes

$$K_i = B^{a'/e_i \bmod (p-1)} \bmod p$$
$$V_i = h(h(h(K_i))).$$

If $V_i$ is the same as $V_B'$ received from Bob, then the guessed password $\pi_i$ is the same as $\pi$, which means that it is correctly guessed:

$$\begin{aligned} K_i &= B^{a'/e_i} \bmod p \\ &= ((f(\pi))^b)^{a'/e_i} \bmod p \\ &= (m^{e_i \cdot b})^{a'/e_i} \bmod p \\ &= m^{a'b} \bmod p \\ &= (A')^b \bmod p \\ &= K'. \end{aligned}$$

To prevent an exponential equivalence attack in SPEKE, Tang and Mitchell suggested that the function $f$ be modified as $f(\pi) = h(\pi||ID_A||ID_B||i) \bmod p$, where $i$ $(i \geq 0)$ is the smallest integer that makes $g$ a generator of a multiplicative subgroup of order $q$ in $\mathbb{Z}_p^*$. However, there is no difference between the function proposed by Tang and Mitchell and the function $f(\pi) = (h(\pi))^2 \bmod p$ in the revised version of SPEKE [7] from the viewpoint of preventing an exponential equivalence attack.

Furthermore, exponential equivalence among hash outputs has no effect on the security of hash-based protocols other than SPEKE or Dragonfly, which use a generator derived from hashing a shared password.

### *3.2. Exponential Equivalence in Dragonfly*

In this subsection, we show the exponential equivalence problem in the Dragonfly protocol. A single on-line attempt can scan through all the exponentially equivalent passwords.

Assume that the outputs of the mapping function $f$ for $j$ passwords from $\pi_1$ to $\pi_j$ are exponentially equivalent with a base $m$. The actual class in Dragonfly is

$$\{f(\text{Alice}, \text{Bob}, \pi_1) = m^{e_1}, f(\text{Alice}, \text{Bob}, \pi_2) = m^{e_2}, \cdots, f(\text{Alice}, \text{Bob}, \pi_j) = m^{e_j}\},$$

where the exponent $e_i$ for each $i \in \{1, 2, ..., j\}$, is an integer greater than or equal to 1. An active attacker, Eve, masquerading as Alice, generates two random $a'_1$ and $a'_2$, and sends $A'$ and $a'$ to Bob when $A' = m^{-a'_1} \bmod p$ and $a' = (a'_1 + a'_2) \bmod q$ are calculated. Bob also generates two random $b_1$ and $b_2$, and sends $B$ and $b$ to Eve when $B = (f(\text{Alice}, \text{Bob}, \pi))^{-b_1} \bmod p$ and $b = (b_1 + b_2) \bmod q$ are calculated. On receiving $A'$ and $a'$ from Eve, Bob computes $K' = ((f(\text{Alice}, \text{Bob}, \pi))^{a'} \cdot A')^{b_2} \bmod p$ and send $V'_B = h(K'||B||b||A'||a')$ to Eve.

On receiving $B$ and $V'_B$ from Bob, Eve terminates the protocol and for each $i$, $1 \leq i \leq j$, computes

$$\begin{aligned} K_i &= (m^b \cdot B^{1/e_i \bmod q})^{(e_i \cdot a' - a'_1) \bmod q \bmod p} \\ V_i &= h(K_i||B||b||A'||a'). \end{aligned}$$

If $V_i$ is the same as $V'_B$ received from Bob, then the guessed password $\pi_i$ is the same as $\pi$, which means that it is correctly guessed:

$$\begin{aligned} K_i &= (m^b \cdot B^{1/e_i})^{(e_i \cdot a' - a'_1)} \bmod p \\ &= (m^b \cdot ((m^{e_i})^{-b_1})^{1/e_i})^{(e_i \cdot a' - a'_1)} \bmod p \\ &= (m^{b_2})^{(e_i \cdot a' - a'_1)} \bmod p \\ &= (m^{(e_i \cdot a' - a'_1)})^{b_2} \bmod p \\ &= ((m^{e_i})^{a'} \cdot A')^{b_2} \bmod p \\ &= K'. \end{aligned}$$

## 4. Prevention of Exponential Equivalence in SPEKE

In this section, we present a simple solution for preventing an attacker from guessing multiple passwords in a single on-line attempt based on exponential equivalence of the passwords in the SPEKE protocol.

Our approach is based on the property that there is no exponentially equivalent element when twice the minimum element is greater than the maximum element within a given set. Similarly, this approach can be applied to SPEKE by modifying the password-mapping function $f$ by prefixing fixed bits to the output of the hash function $h$.

**Theorem 1.** *Let $h : \{0,1\}^* \to \{0,1\}^\ell$ be a function that maps an input of arbitrary length to an output of fixed length $\ell$. Let $p$ be a safe prime, $p = 2q + 1$ where $q$ is also a prime. There is no exponentially equivalent element among all the outputs of $f(x)$ over a subgroup of $\mathbb{Z}_p^*$ of prime order $q$ satisfying*

$$f(x) = (h(x) + k \cdot 2^\ell)^2, \tag{1}$$

*where $k$ is any integer satisfying $3 \le k \le \lfloor \frac{p}{2^\ell} \rfloor - 1$ and $k \neq \lfloor \frac{q-1}{2^\ell} \rfloor$.*

**Proof.** From the given condition $0 \le h(x) < 2^\ell$, we can derive the following equations.

$$k \cdot 2^\ell \le h(x) + k \cdot 2^\ell \le (k+1) \cdot 2^\ell - 1 < (k+1) \cdot 2^\ell. \tag{2}$$

$$k^2 \cdot 2^{2\ell} \le (h(x) + k \cdot 2^\ell)^2 < (k+1)^2 \cdot 2^{2\ell}.$$

$$k^2 \cdot 2^{2\ell} \le f(x) < (k+1)^2 \cdot 2^{2\ell}.$$

For a base $c$, two successive elements of an exponential equivalence class should be $c^i$ and $c^{i+1}$. To ensure that no exponentially equivalent element can be found, the maximum value of $f(x)$ must be less than $c$ times the minimum value for all $c$.

$$c \cdot k^2 \cdot 2^{2\ell} \ge (k+1)^2 \cdot 2^{2\ell}$$

If we divide both sides by $2^{2\ell}$, we get

$$c \cdot k^2 \ge (k+1)^2.$$

Since $c \ge 2$, if $c = 2$ is satisfied, it is satisfied for all $c$:

$$2k^2 > (k+1)^2.$$

$$\therefore k \ge 3 \tag{3}$$

The modular square is symmetric with respect to half of modulus $p$. Therefore, to prevent output duplication, the domain of definition of the modular square, or the range of $h(x) + k \cdot 2^\ell$, must be included in

$$tq + 1 \le h(x) + k \cdot 2^\ell \le (t+1)q, \quad \text{for } t = 0 \text{ or } 1. \tag{4}$$

Because (2) and (4) are the codomain and range of $h(x) + k \cdot 2^\ell$, respectively, the following equation is established:

$$tq + 1 \le k \cdot 2^\ell \text{ and } (k+1) \cdot 2^\ell - 1 \le (t+1)q, \quad \text{for } t = 0 \text{ or } 1.$$

$$\therefore 1 \le k \le \lfloor \frac{q+1}{2^\ell} \rfloor - 1 \text{ or } \lceil \frac{q+1}{2^\ell} \rceil \le k \le \lfloor \frac{p}{2^\ell} \rfloor - 1. \tag{5}$$

From (3) and (5), when (1) is satisfied, no exponentially equivalent element is found in $f(x)$. □

For the 112-bit security level, the sizes of $p$, $q$ and $2^\ell$ should be greater than or equal to 2048, 2047 and 224 bits, respectively [9].

## 5. Discussion

Our solution for preventing exponential equivalence cannot be applied directly over a subgroup in which modulus $p$ is not a safe prime, such as in Dragonfly. To make a generator of a subgroup, exponentiation with exponent $r$, *i.e.*, $f(x) = (h(x) + k \cdot 2^{\ell})^r$, is needed. Because $r$ is very large, e.g., an 896-bit integer corresponding to a 1024-bit modulus, subsequently most of outputs of $f(x)$ will exceed $p$. We reserve this problem for future research.

## 6. Conclusions

In this paper, we showed that an active attacker can try multiple passwords in a single attempt by exploiting the exponential equivalence of the passwords in Dragonfly as well as in SPEKE. Furthermore, we proposed and proved a simple solution for preventing this problem in the SPEKE protocol. Our solution can be easily applied to variants of the Diffie-Hellman protocol (e.g., the B-SPEKE protocol) that use a safe prime modulus and do not use a fixed generator.

## Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT, and Future Planning (2014R1A1A2002775).

## Author Contributions

Hanwook Lee conceived the idea and wrote this manuscript. Dongho Won contributed to the direction and content and also revised the manuscript.

## Conflicts of Interest

The authors declare no conflict of interest.

## References

1. Jablon, D. Strong password-only authenticated key exchange. *Comput. Commun. Rev. ACM SIGCOMM* **1996**, 26, 5–26.
2. Harkins, D. Simultaneous authentication of equals: A secure, password-based key exchange for mesh networks. In Proceedings of the Second International Conference on Sensor Technologies and Applications, Cap Esterel, France, 25–31 August 2008; pp. 839–844.
3. Zhang, M. Analysis of the SPEKE password-authenticated key exchange protocol. *IEEE Commun. Lett.* **2004**, *8*, 63–65.
4. Tang, Q.; Mitchell, C.J. On the Security of Some Password-Based Key Agreement Schemes. In *Computational Intelligence and Security*; Springer Berlin Heidelber: Berlin, Germany, 2005; pp. 149–154.

5. Hao, F.; Shahandashti, S.F. The SPEKE Protocol Revisited. In *Security Standardisation Research*; Springer International Publishing: Cham, Switzerland, 2014; pp. 26–38.
6. Clarke, D.; Hao, F. Cryptanalysis of the Dragonfly key exchange protocol. *IET Inf. Secur.* **2010**, *8*, 283–289.
7. Jablon, D. Extended password key exchange protocols immune to dictionary attack. In Proceedings of the Sixth IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Cambridge, MA, USA, 18–20 June 1997; pp. 248–255.
8. Hao, F.; Ryan, P. J-PAKE: Authenticated key exchange without PKI. In *Transactions on computational science XI*; Springer Berlin Heidelberg: Berlin, Germany, **2010**, 192–206.
9. Barker, E.; Roginsky, A. NIST Special Publication 800-131A Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths. Available online: http://csrc.nist.gov/publications/nistpubs/800-131A/sp800-131A.pdf (accessed on 1 September 2015).