New Approaches to Password Authenticated Key Exchange based on RSA

Muxiang Zhang
Verizon Communications, Inc.
40 Sylvan Road, Waltham, MA 02451, USA
muxiang.zhang@verizon.com

February 4, 2004

Abstract

We investigate efficient protocols for password-authenticated key exchange based on the RSA public-key cryptosystem. To date, most of the published protocols for password-authenticated key exchange were based on Diffie-Hellman key exchange. It appears inappropriate to design password-authenticated key exchange protocols using RSA and other public-key cryptographic techniques. In fact, many of the proposed protocols for password-authenticated key exchange based on RSA have been shown to be insecure; the only one that remains secure is the SNAPI protocol. Unfortunately, the SNAPI protocol has to use a prime public exponent e larger than the RSA modulus n. In this paper, we present a new password-authenticated key exchange protocol, called *PEKEP*, which allows using both large and small prime numbers as RSA public exponents. Based on number-theoretic techniques, we show that the new protocol is secure against the *e-residue attack*, a special type of off-line dictionary attack against RSA-based password-authenticated key exchange protocols. We also provide a formal security analysis of PEKEP under the RSA assumption and the random oracle model. On the basis of PEKEP, we present a computationally-efficient key exchange protocol to mitigate the burden on communication entities.

Key words: password authentication, off-line dictionary attack, public-key cryptography

1 Introduction

The design of authentication and key exchange protocol is usually based on the assumption that entities either share or own some secret data (called keys) which are drawn from a space so large that an adversary can not enumerate, either on-line or off-line, all possible keys in the space. In practice, however, cryptographic keys may often be substituted by human-memorable passwords consisting of only six to ten characters. The consequence is the proliferation of the so-called exhaustive guessing or dictionary attacks against many password-based systems [27]. Password guessing attacks have been around for so long, it seems paradoxical that strong authentication using only small passwords would be possible. In 1992, Bellovin and Merritt [2] showed that such paradoxical protocols did exist. They presented a family of protocols known as *Encrypted Key Exchange*, or *EKE*. By using a combination of symmetric and asymmetric (public-key) cryptographic techniques, EKE provides insufficient information for an adversary to verify a guessed password and thus defeats off-line dictionary attacks. Following EKE, a number of protocols for password-based authentication and key exchange have been proposed, e.g., [3-5, 10-11, 15, 17-19, 26]. A comprehensive list of such protocols can be found in Jablon's research link [16].

Password-authenticated key exchange protocols are attractive for their simplicity, convenience, and strength against off-line dictionary attacks. Unlike other public-key based key exchange protocols such as SSL, the EKE-like protocols do not rely on the existence of a public key infrastructure (PKI). This is desirable in many environments where the deployment of a public key infrastructure is either not possible or would be overly complex. Over the last decade, many researchers have investigated the feasibility of implementing EKE using different public-key cryptosystems such as RSA, ElGamal, and Diffie-Hellman key exchange. Nonetheless, most of the well-known and secure variants of EKE are based on Diffie-Hellman key-exchange. It seems that EKE works well with Diffie-Hellman key exchange, but presents subtleties one way or the other when implemented with RSA and other public-key cryptosystems. In fact, many of the proposed protocols for password-authenticated key exchange based on RSA have been shown to be insecure [2, 24, 22]; the only one that remains secure is the SNAPI protocol developed by Mackenzie, et al. [22]. Unfortunately, the SNAPI protocol has to use a prime public exponent e which is larger than the RSA modulus n. This may render the SNAPI protocol impractical in resource-limited platforms, such as mobile phones and personal digital assistants.

In this paper, we investigate the design of RSA-based password-authenticated key exchange protocols that do not restrict the size of RSA public exponent. We focus on such protocols in the two-party setting, where two entities (say, Alice and Bob) share a password drawn from a small space and one of the entities (say, Alice) possesses a pair of RSA public/private keys. A nice feature of this type of protocols is that the overhead for the protocol setup is minimal; Alice and Bob only need to establish a shared password *in advance* and do not need to establish other common parameters such as a prime number p and a generator of the cyclic group modulo p. This is appealing in environments where entities have insufficient resources to generate or validate public parameters with certain properties, e.g., primality.

1.1 Related Work

In 1989, Lomas et al. [20] introduced the first password-authenticated key exchange protocol resistant to off-line dictionary attacks. Their protocol, however, assumed that one of the entities knows the other entity's public key and thus is not a strictly password-only protocol. This type of protocol was further studied by Gong [12] and by Halevi and Krawczyk [13].

The EKE protocol developed by Bellovin and Merritt is the first password-authenticated key exchange protocol that does not require one of the entities (say, Bob) to know the public key of the other entity (say, Alice). Following EKE, many researchers have provided a variety of extensions to achieve additional goals, e.g., protection against server compromise [3, 19, 26]. In their original paper [2], Bellovin and Merritt investigated the feasibility of implementing EKE using three different types of public-key cryptographic techniques: RSA, ElGamal, and Diffie-Hellman key exchange. They pointed out that EKE is only suitable for implementation using Diffie-Hellman key exchange. The problem with the EKE variant implemented using RSA is that Bob does not know the public key, (n, e), of Alice and thus can not verify whether e is relatively prime to $\phi(n)$. This fosters the so-called e-residue attack as described in [2]. Based on number-theoretic techniques, Patel [24] further investigated the security of the RSA-based EKE variant and concluded that simple modifications of EKE would not prevent the RSA-based EKE variant from off-line dictionary attacks. In 1997, Lucks [21] proposed an RSA-based password-authenticated key exchange protocol (called OKE) which was claimed to be secure against the e-residue attack. Later, Mackenzie et al. [22] found that the OKE protocol is still subject to the e-residue attack. In [22], Mackenzie et al. proposed an RSA-based password-authenticated key exchange protocol (SNAPI) and provided a formal security proof in the random oracle model. The SNAPI protocol mandates that the public exponent e be a prime number larger than the RSA modulus n. This ensures that e is relatively prime to $\phi(n)$.

To avoid using large public exponents, Zhu et al. [28] proposed an "interactive" protocol which is revised from an idea of [2]. In the interactive protocol, Bob sends to Alice a number of messages encrypted using Alice's public key. If Alice can successfully decrypt the encrypted messages, then Bob is ensured that the encryption based on Alice's public key is a permutation. A drawback of the interactive protocol is the large communication overhead involved in the verification of Alice's public key.

1.2 Overview of Our Results

In this paper, we present a new protocol for password-authenticated key exchange based on RSA. The new protocol, called *PEKEP*, involves two entities, Alice and Bob, who share a short password and Alice possesses a pair of RSA keys, n, e and d, where $ed \equiv 1 \pmod{\phi(n)}$. Unlike the protocol SNAPI, however, the new protocol PEKEP allows Alice to select both large and small primes for the RSA public exponent e, In the protocol PEKEP, Bob does not need to verify if e is relatively prime to $\phi(n)$, and furthermore, Bob may even not test the primality for a large public exponent selected by Alice. When e is a prime number larger than n, we show that the protocol PEKEP is nearly identical to the protocol SNAPI except minor differences. Based on number-theoretic techniques, we prove that the protocol PEKEP is secure against the e-residue attack as described in [2, 24]. We also provide a formal security analysis of PEKEP under the RSA assumption and the random oracle model.

To reduce the computational load on entities, we present a computationally-efficient key exchange protocol (called *CEKEP*) in this paper. The protocol CEKEP is derived from PEKEP by adding two additional flows between Alice and Bob. With the two additional flows, we show that the probability for an adversary to launch a successful e-residue attack against CEKEP is less than or equal to ε, where ε is a small number (e.g., $0 < \varepsilon \leq 2^{-80}$) selected by Bob. In the protocol CEKEP, the computational burden on Bob includes two modular exponentiations, each having an exponent of $\mathcal{O}(\lceil \log_2 \varepsilon^{-1} \rceil)$ bits. When $\varepsilon = 2^{-80}$, for example, the computational burden on Bob is lighter than that in a Diffie-Hellman based password-authenticated key exchange protocol. In the Diffie-Hellman based EKE variant, Bob needs to compute two modular exponentiations, each having an exponent of at least 160 bits. We show that the computational burden on Bob could be reduced further if Bob caches a hashed version of Alice's public key used in previous communication sessions.

The rest of the paper is organized as follows. In Section 2, we review basic concepts of number theory used throughout this paper. We provide an overview of the security model for password-authenticated key exchange in Section 3. We present the protocol PEKEP in Section 4 and investigate its security against e-residue attacks. We present the protocol CEKEP in Section 5 and pursue formal security analysis of PEKEP and CEKEP in Section 6. We conclude in Section 7.

2 Preliminaries

Let $\{0,1\}^k$ denote the set of binary strings of length k and $\{0,1\}^*$ denote the set of binary strings of finite length. Without confusion, we sometimes use s_1, s_2 to denote the concatenation of two strings s_1 and s_2. A real-valued function $\epsilon(k)$ of non-negative integers is called *negligible* (in k) if for every $c > 0$, there exists $k_0 > 0$ such that $\epsilon(k) \leq 1/k^c$ for all $k > k_0$.

For every positive integer n, $n > 1$, it is well know that n can be expressed as a product of nontrivial powers of distinct primes, i.e., $n = p_1^{a_1} p_2^{a_2} \ldots p_r^{a_r}$, where $p_1, p_2, \ldots, p_r$ are primes and $a_1, a_2, \ldots, a_r$ are positive integers. Up to a rearrangement of the prime powers, this *prime-power factorization* is unique. Let $\mathbb{Z}_n$ denote the set of non-negative integers less than n and $\mathbb{Z}_n^*$ denote the set consisting of integers in $\mathbb{Z}_n$ that are relatively prime to n. The number of integers in $\mathbb{Z}_n^*$ is equal to the *Euler phi-function* $\phi(n)$.

Let a, b, and n be integers such that $n > 0$ and $\gcd(a, n) = c$. If $c \nmid b$, then the congruence $ax \equiv b \pmod{n}$ has no solutions. If $c \mid b$,the congruence $ax \equiv b \pmod{n}$ has exactly c incongruent solutions modulo n. Let x_0 denote one of the solutions, then the c incongruent solutions are given by

$$x = x_0 + t(n/c), \quad t = 0, 1, \ldots, c-1. \tag{1}$$

Let g and n be positive integers relatively prime to each other. The least positive integer i such that $g^i \equiv 1 \pmod{n}$ is called the *order of g modulo n*. If the order of g is equal to $\phi(n)$, then g is called a *primitive root of n*. It is known (see [1, 25]) that a positive integer n, $n > 1$, possesses a primitive root if and only if $n = 2, 4, p^t$ or $2p^t$, where p is an odd prime and t is a positive integer. If n has a primitive root g, then the integers $g^0, g^1, g^2, \ldots, g^{\phi(n)-1}$ form a cyclic group under the modulo n multiplication. Due to this fact, we see that if a is a positive integer relatively prime to n, then there exists a unique integer i, $0 \leq i \leq \phi(n) - 1$, such that $a = g^i \bmod n$. The integer i is called the *index of a to the base g modulo n*, and is denoted by $\mathrm{ind}_g a$. With this notation, we have $a = g^{\mathrm{ind}_g a} \bmod n$.

If n and e are positive integers and a is an integer relatively prime to n, then we say that a is an *e-th power residue of n* if the congruence $x^e \equiv a \pmod{n}$ has a solution. If n has a primitive root, then a is an e-th power residue of n if and only if $a^{\phi(n)/b} \equiv 1 \pmod{n}$, where $b = \gcd(e, \phi(n))$.

3 Security Model

We consider two-party protocols for authenticated key-exchange using human-memorable passwords. In its simplest form, such a protocol involves two entities, say *Alice* and *Bob* (denoted by A and B), both possessing a secret password drawn from a small password space $\mathcal{D}$. Based on the password, Alice and Bob can authenticate each other and upon a successful authentication, establish a session key which is known to nobody but the two of them. There is present an active adversary, denoted by $\mathcal{A}$, who intends to defeat the goal for the protocol. The adversary has full control of the communications between Alice and Bob. She can deliver messages out of order and to unintended recipients, concoct messages of her own choosing, and create multiple instances of entities and communicate with these instances in parallel sessions. She can also enumerate, *off-line*, all the passwords in the password space $\mathcal{D}$. She can even acquire session keys of accepted entity instances. Our formal model of security for password-authenticated key exchange protocols is based on that of [5]. In the following, we review the operations of the adversary and formulate the definition of security. For details as well as motivations behind the model, we refer the readers to [5].

Initialization. Let I denote the identities of the protocol participants. Elements of I will often be denoted A and B (Alice and Bob). We emphasis that A and B are variables ranging over I and not fixed members of I. Each pair of entities, $A, B \in I$, are assigned a password w which is randomly selected from the password space $\mathcal{D}$. The initialization process may also specify a set of cryptographic functions (e.g., hash functions) and establishes a number of cryptographic parameters.

Running the Protocol. Mathematically, a protocol Π is a probabilistic polynomial-time algorithm which determines how entities behave in response to received input. For each entity, there may be multiple instances running the protocol in parallel. We denote the i-th instance of entity A as Π_A^i. The adversary $\mathcal{A}$ can make queries to any instance; she has an endless supply of Π_A^i oracles ($A \in I$ and $i \in \mathbb{N}$). In response to each query, an instance updates its internal state and gives its output to the adversary. At any point in time, the instance may accept and possesses a session key sk, a session id sid, and a partner id pid. The query types, as defined in [5], include:

- Send(A, i, M): This sends message M to instance Π_A^i. The instance executes as specified by the protocol and sends back its response to the adversary. Should the instance accept, this fact, as well as the session id and partner id will be made visible to the adversary.

- Execute(A, i, B, j): This call carries out an honest execution between two instances Π_A^i and Π_B^j, where $A, B \in I, A \neq B$ and instances Π_A^i and Π_B^j were not used before. At the end of the execution, a transcript is given to the adversary, which logs everything an adversary could see during the execution (for details, see [5]).

- Reveal(A, i): The session key sk_A^i of Π_A^i is given to the adversary.

- Test(A, i): The instance Π_A^i generates a random bit b and outputs its session key sk_A^i to the adversary if $b = 1$, or else a random session key if $b = 0$. This query is allowed only once, at any time during the adversary's execution.

- Oracle(M): This gives the adversary oracle access to a function h, which is selected at random from some probability space Ω. The choice of Ω determines whether we are working in the standard model, or in the random-oracle model (see [5] for further explanations).

In additional to the above query types, we introduce another query type:

- Impersonate$(A, i, \pi, paras)$: This replaces the password and the parameters of the instance Π_A^i by π and $paras$, respectively, where Π_A^i was not used before. After this query, the internal state of Π_A^i is visible to the adversary. Each query of this type is also called an *impersonation attempt*.

We use the Impersonate type to model an impersonation attack, which allows the adversary to test a guessed password *on-line*. In an impersonation attack, the adversary picks a password π as her guess and then impersonates as an instance Π_A^i to start the protocol towards another instance Π_B^j. By observing the decision of Π_B^j (i.e., accepts or rejects), the adversary tests the correctness of the guessed password π. Furthermore, by analyzing, off-line, the transcript of the execution, the adversary may be able to test passwords other than π. For a secure protocol, we expect that the adversary can only test a single password in each impersonation attempt. Certainly, the impersonation attack can be implemented by solely using the Send query type. The number of Send queries called by the adversary, however, may vary with different protocols. Using the Impersonate type, we can *explicitly* defines the number of impersonation attempts performed by the adversary. We assume that the adversary always use an impersonated instance to launch an impersonation attack.

DEFINITIONS. Let Π_A^i and Π_B^i, $A \neq B$, be a pair of instances. We say that Π_A^i and Π_B^i are *partnered* if both instances have accepted and hold the same session id sid and the same session key sk. Here, we define the sid of Π_A^i (or Π_B^i) as the concatenation of all the messages sent and received by Π_A^i (or Π_B^i). We say that Π_A^i is *fresh* if: i) it has accepted; ii) it is not impersonated; and iii) a Reveal query has not been called either on Π_A^i or on its partner. With these notions, we now define the advantage of the adversary $\mathcal{A}$ in attacking the protocol. Let Succ denote the event that $\mathcal{A}$ asks a single Test query on a fresh instance, outputs a bit b', and $b' = b$, where b is the bit selected during the Test query. The advantage of the adversary $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^{ake} = 2Pr(\mathsf{Succ}) - 1$.

Definition 1 *A protocol Π is called a secure password-authenticated key exchange protocol if for every polynomial-time adversary $\mathcal{A}$ that makes at most v impersonation attempts, the following two conditions are satisfied:*

1) *Except with negligible probability, each oracle call* Execute(A, i, B, j) *produces a pair of partnered instances Π_A^i and Π_B^j.*

2) $\mathsf{Adv}_{\mathcal{A}}^{ake} \leq v/|\mathcal{D}| + \epsilon$, *where* $|\mathcal{D}|$ *denotes the size of the password space and* ϵ *is a negligible function.*

The first condition basically says that when the adversary carries out an honest execution between two instances Π_A^i and Π_B^j $(A \neq B)$, both instances accept and hold the same session key and the same session id. To explain the second condition, let E_v denote the event the adversary makes (at most) v impersonation attempts and one of them obtains a correct guess of the password of A (or B). When E_v is true, the adversary's success probability in attacking the protocol could be 1. When E_v is false (denoted by $\neg E_v$), we expect that the adversary's success probability in attacking a *secure* protocol is bounded by $0.5 + \epsilon'$, where ϵ' is negligible. Hence, for a secure protocol Π, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{ake} &= 2Pr(\mathsf{Succ}) - 1 \\
&= 2Pr(\mathsf{Succ}|_{E_v})Pr(E_v) + 2Pr(\mathsf{Succ}|_{\neg E_v})Pr(\neg E_v) - 1 \\
&\leq \frac{2v}{|\mathcal{D}|} + 2(0.5 + \epsilon')(1 - \frac{v}{|\mathcal{D}|}) - 1 \\
&= \frac{v}{|\mathcal{D}|} + \epsilon
\end{aligned}$$

where $\epsilon = 2\epsilon'(1 - v/|\mathcal{D}|)$. So, the conditions of Definition 1 capture our expectation for a secure password-authenticated key exchange protocol.

4 Password Enabled Key Exchange Protocol

In this section, we present a new protocol, called *Password Enabled Key Exchange Protocol*, or simply, *PEKEP*. It is an RSA-based password-authenticated key exchange protocol, but it allows using both large and small prime numbers as RSA public exponents. Let A and B denote the identities of Alice and Bob who share a password w drawn from a password space $\mathcal{D}$. Alice has also generated a pair of RSA keys n, e and d, where n is a large positive integer (e.g., $n > 2^{1023}$) equal to the product of two primes of (roughly) the same size, e is a positive integer relatively prime to $\phi(n)$, and d is a positive integer such that $ed \equiv 1 \pmod{\phi(n)}$. The encryption function is define by $E(x) = x^e \bmod n$, $x \in \mathbb{Z}_n$. The decryption function is $D(x) = x^d \bmod n$. For a positive integer m, we define E^m recursively as

$$E^m(x) = E(E^{m-1}(x)) = x^{e^m} \mod n.$$

Likewise,

$$D^m(x) = D(D^{m-1}(x)) = x^{d^m} \mod n.$$

Define hash functions $H_1, H_2, H_3 : \{0,1\}^* \to \{0,1\}^k$ and $H : \{0,1\}^* \to \mathbb{Z}_n$, where k is a security parameter, e.g., $k = 160$. Note that H can be implemented using a standard hash function $h : \{0,1\}^* \to \{0,1\}^\ell$, where ℓ is the length of n, i.e., $\ell = \lceil \log_2 n \rceil$. On input x, $H(x) = h(x)$, if $h(x) < n$, and $H(x) = h(x) - \lceil n/2 \rceil$ if else. Assume that h is a random function, then for any integer $b \in \mathbb{Z}_n$, it can be proved that $|Pr(H(x) = b) - \frac{1}{n}| < 2^{-\ell}$; the bias is negligible. In the following, we assume that H_1, H_2, H_3 and H are independent random functions.

The protocol PEKEP is described in Fig. 1. Alice starts the protocol by sending her public key (n, e) and a random number $r_A \in_R \{0,1\}^k$ to Bob. Bob verifies if e is an odd prime and n is an odd integer. If not, Bob rejects; otherwise, Bob computers an integer $m = \lfloor \log_e n \rfloor$ and selects two random numbers $a \in_R \mathbb{Z}_n^*$, and $r_B \in_R \{0,1\}^k$. Bob then computes $\alpha = H(w, r_A, r_B, A, B, n, e)$ and checks if $\gcd(\alpha, n) = 1$. If $\gcd(\alpha, n) \neq 1$, Bob assigns a random number of $\mathbb{Z}_n^*$ to λ; otherwise, Bob assigns α to λ. Next, Bob computes $z = E^m(\lambda E(a))$ and sends r_B and z to Alice. Subsequently, Alice computes α using her password w and checks if α and n are relatively prime. If $\gcd(\alpha, n) \neq 1$,

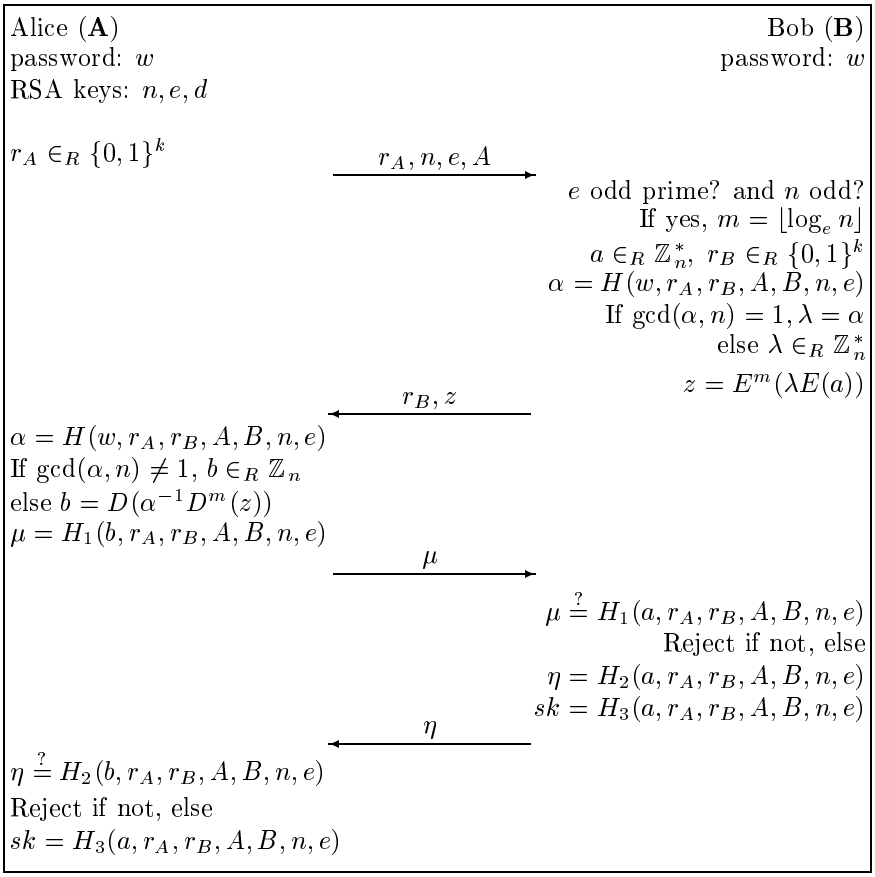

Figure 1: Password Enabled Key Exchange Protocol (PEKEP)

Alice assigns a random number of $\mathbb{Z}_n$ to the variable b. If $\gcd(\alpha, n) = 1$ and z is an e^m-th power residue of n, Alice sets $b = D(\alpha^{-1} D^m(z))$. Next, Alice and Bob authenticate each other using a and b and generate a session key sk upon successful authentication.

In the protocol PEKEP, both Alice and Bob intend to reject when they detect the event $\gcd(\alpha, n) \neq 1$. To avid leaking any information about this event, Alice and Bob use random numbers to compute their responses μ, z, and η. If $n = pq$ is the product of two large primes of about the same size, then the probability of such an event is negligible. When $\gcd(\alpha, n) = 1$, Alice and Bob agree on a secret number $b = a$ and thus can use the secret number to authenticate each other and establish a shared session key.

Note that, when e is a prime number larger than n, Bob sets $m = 0$. In this case, the run of PEKEP is nearly identical to that of SNAPI. A minor difference between the two protocols is that, in SNAPI, Alice rejects immediately when she detects the event $\gcd(\alpha, n) \neq 1$; while in PEKEP, Alice sends a random number as her response to Bob and expects Bob to reject. As mentioned above, the event $\gcd(\alpha, n) \neq 1$ occurs with negligible probability if $n = pq$ is the product of two large primes of about the same size. In the protocol PEKEP, Bob only verifies if the public exponent e is an odd prime and the RSA modulus n is an odd integer; Bob does not verify if e is relatively prime to $\phi(n)$. This may foster the so-called e-residue attack as described in [2, 24]. In the e-residue attack, an adversary, say, *Eva*, selects $\pi_0 \in \mathcal{D}$ as her guess of Alice's password. She also selects an odd prime number e and an odd integer n such that $e \mid \phi(n)$, i.e., (n, e) is not a valid RSA public key. Then Eva impersonates as Alice and starts the protocol PEKEP by sending r_E, n, e, A to Bob,

where $r_E \in \{0,1\}^k$ is a random number generated by Eva. After receiving r_B and z from Bob, Eva Computes μ and sends it back to Bob. If Bob accepts, then Eva has a successful guess of Alice's password (i.e., π_0). If Bob rejects, on the other hand, Eva excludes her guess π_0 from the password space $\mathcal{D}$. Furthermore, Eva may exclude more passwords by repeating, *off-line*, the following three steps:

1) Eva selects a password π from $\mathcal{D}$.

2) Eva computes $\alpha = H(\pi, r_E, r_B, A, B, n, e)$.

3) Eva tests if $\gcd(\alpha, n) = 1$. If not, Eva returns to step 1; otherwise, Eva verifies if the congruence $(\alpha x^e)^{e^m} \equiv z \pmod{n}$ has a solution in $\mathbb{Z}_n^*$. If the congruence has a solution, Eva returns to step 1. If the congruence has no solution in $\mathbb{Z}_n^*$, then Eva knows that π is not the password of Alice. Next, Eva excludes π from $\mathcal{D}$ and returns to step 1.

We say that Eva succeeds if she can exclude more than one password in the e-residue attack as described above. In the following, we show that the protocol PEKEP is secure against e-residue attacks.

Theorem 1 *Let n, $n > 1$, be an odd integer with prime-power factorization $n = p_1^{a_1} p_2^{a_2} \dots p_r^{a_r}$. Let m be a non-negative integer and e an odd prime such that for any prime-power $p_i^{a_i}$ of the factorization of n, $e^{m+1} \nmid \phi(p_i^{a_i}), 1 \le i \le r$. If z is an e^m-th power residue of n, then for any $\lambda \in \mathbb{Z}_n^*$, the congruence $(\lambda x^e)^{e^m} \equiv z \pmod{n}$ has a solution in $\mathbb{Z}_n^*$.*

Proof. To prove that $(\lambda x^e)^{e^m} \equiv z \pmod{n}$ has a solution in $\mathbb{Z}_n^*$, we only need to prove that, for each prime power $p_i^{a_i}$ of the factorization of n, the following congruence

$$(\lambda x^e)^{e^m} \equiv z \pmod{p_i^{a_i}} \tag{2}$$

has a solution in $\mathbb{Z}_{p_i^{a_i}}^*$.

Let $n_i = p_i^{a_i}, 1 \le i \le r$. Then $\phi(n_i) = p_i^{a_i - 1}(p_i - 1)$. Since n is odd, p_i is an odd prime. Thus, the integer n_i possesses a primitive root. Let g be a primitive root of n_i, that is, $g^{\phi(n_i)} = 1 \bmod n_i$, and for any $0 \le i, j \le \phi(n_i) - 1, i \ne j$, $g^i \ne g^j \bmod n_i$. Let $\gcd(e^m, \phi(n_i)) = e^c, 0 \le c \le m$. We consider the following two cases:

(1) If $c = 0$, then e and $\phi(n_i)$ are relatively prime. In this case, it is clear that the congruence $(\lambda x^e)^{e^m} \equiv z \pmod{n_i}$ has a unique solution in $\mathbb{Z}_{n_i}^*$.

(2) Next, we consider the case that $1 \le c \le m$. Since z is an e^m-th power residue of n, the congruence $y^{e^m} \equiv z \pmod{n}$ has solutions in $\mathbb{Z}_n^*$. By the Chinese Remainder Theorem, the following congruence

$$y^{e^m} \equiv z \pmod{n_i} \tag{3}$$

has solutions in $\mathbb{Z}_{n_i}^*$. Let $\mathrm{ind}_g z$ denote the index of z to the base g modulo n_i and let $y \in \mathbb{Z}_{n_i}^*$ be a solution of (3), then

$$g^{e^m \mathrm{ind}_g y - \mathrm{ind}_g z} \equiv 1 \pmod{n_i}.$$

Since the order of g modulo n_i is $\phi(n_i)$, we have

$$e^m \mathrm{ind}_g y \equiv \mathrm{ind}_g z \pmod{\phi(n_i)} \tag{4}$$

Also since $\gcd(e^m, \phi(n_i)) = e^c$, equation (4) has exactly e^c incongruent solutions modulo $\phi(n_i)$ when taking $\mathrm{ind}_g y$ as variable. This indicates that equation (3) has e^c solutions in $\mathbb{Z}_{n_i}^*$. Let y_0 denote one of the solutions of (3), by (1), the e^c incongruent solutions of (4) are given by

$$\mathrm{ind}_g y = \mathrm{ind}_g y_0 + t\phi(n_i)/e^c \pmod{\phi(n_i)}, \quad 0 \le t \le e^c - 1. \tag{5}$$

For any $\lambda \in \mathbb{Z}_n^*$, we have

$$\mathrm{ind}_g y - \mathrm{ind}_g \lambda \equiv \mathrm{ind}_g y_0 - \mathrm{ind}_g \lambda + t\phi(n_i)/e^c \pmod{\phi(n_i)}, \quad 0 \le t \le e^c - 1.$$

Under the condition that $e^{m+1} \nmid \phi(n_i)$, it is clear that $e \nmid \phi(n_i)/e^c$. Hence, $\phi(n_i)/e^c \equiv 1 \pmod{e}$. So, there exists an integer t, $0 \le t \le e-1$, such that

$$\mathrm{ind}_g y_0 - \mathrm{ind}_g \lambda + t\phi(n_i)/e^c \equiv 0 \pmod{e},$$

which implies that there exists an integer $y \in \mathbb{Z}_{n_i}^*$, such that $y^{e^m} \equiv z \pmod{n_i}$ and $y\lambda^{-1}$ is an e-th power residue of n_i. Therefore, equation (2) has a solution in $\mathbb{Z}_{n_i}^*$, which proves the theorem. □

In PEKEP, Bob sets m equal to $\lfloor \log_e n \rfloor$. Thus, for every prime-power $p_i^{a_i}$ of the factorization of n, we have $e^{m+1} > n \ge p_i^{a_i}$. By Theorem 1, for any $\lambda \in \mathbb{Z}_n^*$, the congruence $(\lambda x^e)^{e^m} \equiv z \pmod{n}$ has a solution in $\mathbb{Z}_n^*$, where z is the e-th power residue computed by Bob. Hence, by repeating (off-line) the three steps as described previously, Eva could not exclude any password from the space $\mathcal{D}$. So, the protocol PEKEP is secure against e-residue attacks. In Section 6, we will provide a formal analysis of PEKEP within the security model described in Section 3.

At the beginning of PEKEP, Bob needs to test the primality of the public exponent e selected by Alice. When e is small, e.g., $e = 17$, the primality test only induces moderate overhead on Bob. When e is large (e.g., $e > 2^{512}$), however, the computational load for the primality test would be tremendous. In the following, we show that primality test of large public exponents by Bob could be avoided with slight modification of PEKEP. In the protocol PEKEP, Bob can actually select a small prime number e' (e.g., $e' = 3$) and replaces Alice's public key (n, e) by (n, e'), that is, Bob computes m, α, z, η, sk using (n, e') instead of Alice's public key (n, e). Theorem 1 demonstrates that the replacement does not lead to e-residue attacks, even if e' is not relatively prime to $\phi(n)$. So, when the public exponent e received from Alice exceeds a threshold, Bob replaces e by a smaller prime number e' $(2 < e' < e)$ of his own choosing. Bob sends r_B, z, and e' to Alice in the second flow. After receiving e' from Bob, Alice tests if e' is relatively prime to $\phi(n)$. If $\gcd(e', \phi(n)) \neq 1$, Alice sends a random number $\mu \in \{0,1\}^k$ to Bob; Alice may select a smaller prime number for e in the next communication session. If $\gcd(e', \phi(n)) = 1$, Alice replaces her decryption key by d' and then proceeds as specified in Fig. 1, where $e'd' \equiv 1 \pmod{\phi(n)}$. If n is the product of two safe primes p and q, that is, $(p-1)/2$ and $(q-1)/2$ are also prime numbers, then for every odd prime number e' less than $(p-1)/2$ and $(q-1)/2$, e' is always prime to $\phi(n)$.

In each run of PEKEP, Bob computes $m+1$ encryptions using Alice's public key (n, e), where $m = \lfloor \log_e n \rfloor$. The computation time for the $m+1$ encryptions is $\mathcal{O}((\log_2 n)^3)$, which means that the computational load on Bob is about the same as that in SNAPI. Since Alice has knowledge of $\phi(n)$, she only needs to perform two decryptions in each run of PEKEP; one using the decryption key $d_1 = d$ and another using the decryption key $d_2 = d^m \bmod \phi(n)$. Note that the computational load on Bob is high even when e is small. In Section 5, we present a computationally-efficient key exchange protocol which greatly reduces the computational load on Bob.

5 Computationally-Efficient Key Exchange Protocol

In this section, we present a *Computationally-Efficient Key Exchange Protocol* (CEKEP), which is described in Fig. 2. The protocol CEKEP is based on PEKEP, but the number of encryptions performed by Bob is less than $\lfloor \log_e n \rfloor$, where (n, e) is the public key of Alice. In the protocol CEKEP, Bob possesses a small number ε, $0 < \varepsilon \le 2^{-80}$, which determines the probability of a successful e-residue attack against the protocol CEKEP. Alice starts the protocol CEKEP by sending her public key n, e and two random numbers $\rho, r_A \in_R \{0,1\}^k$ to Bob. Bob verifies if e is an odd prime and

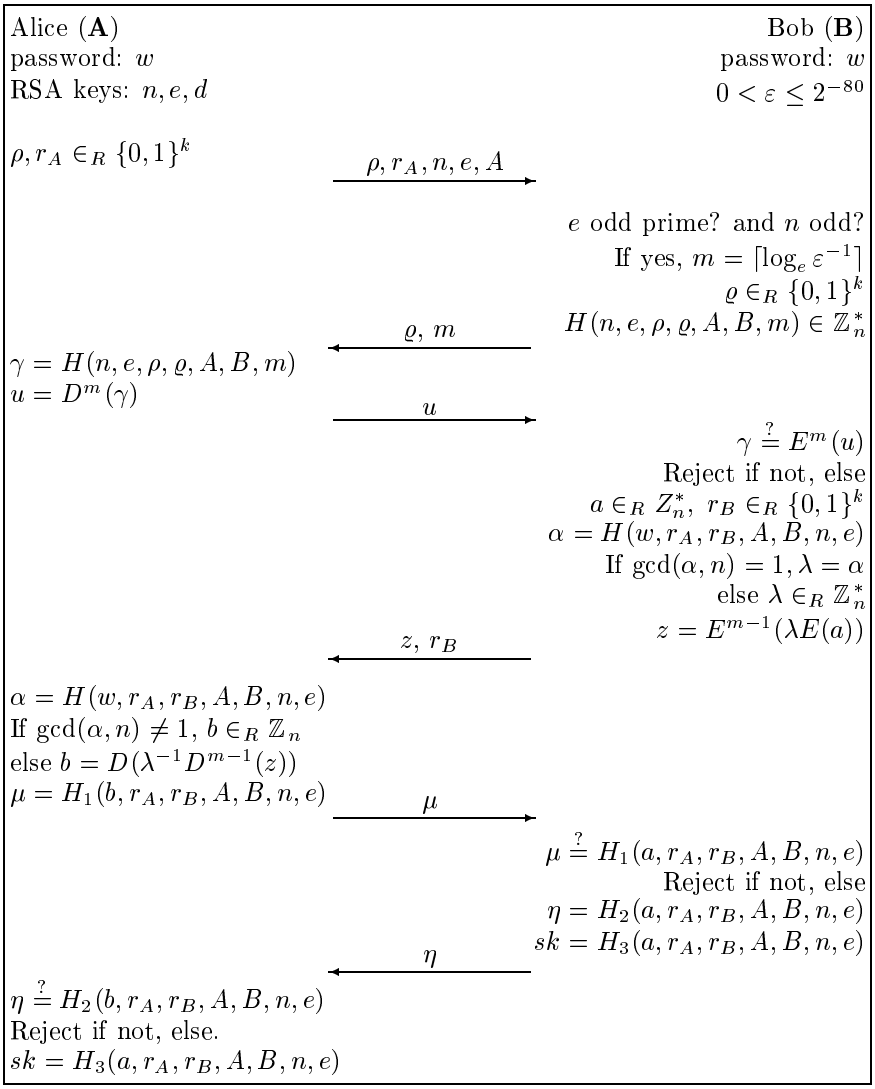

Figure 2: Computationally-Efficient Key Exchange Protocol (CEKEP)

n is an odd integer. If not, Bob rejects. Else, Bob computers an integer m based on e and ε as $m = \lceil \log_e \varepsilon^{-1} \rceil$. Then Bob selects a random number $\varrho \in_R \{0,1\}^k$ such that $\gamma = H(n, e, \rho, \varrho, A, B, m)$ is relatively prime to n. Bob sends ϱ and m to Alice. After receiving ϱ and m, Alice computes $u = D^m(\gamma)$ and sends it back to Bob. Subsequently, Bob verifies if Alice has made the right decryption, i.e., $E^m(u) = \gamma$. If $\gamma \neq E^m(u)$, Bob rejects. Else, Alice and Bob executes the rest of the protocol as in PEKEP.

A major difference between CEKEP and PEKEP is that the protocol CEKEP adds two additional flows between Alice and Bob. Through the two flows, Alice and Bob establish a random number $\gamma \in \mathbb{Z}_n^*$. Then Alice decrypts the random number γ repeatedly m times. If the m repeated decryption is correct, i.e., $\gamma = E^m(u)$, then it can be concluded that, except with probability as small as e^{-m}, the integer e^m does not divide $\phi(p_i^{a_i})$ for every prime-power $p_i^{a_i}$ of the factorization of n.

Theorem 2 *Let n, $n > 1$, be an odd integer with prime-power factorization $n = p_1^{a_1} p_2^{a_2} \ldots p_r^{a_r}$. Let m be a positive integer and e an odd prime. If there exists a prime power, say $p_i^{a_i}$, of the factorization of n such that $e^m \mid \phi(p_i^{a_i})$, then for an integer γ randomly selected from $\mathbb{Z}_n^*$, the probability that γ is*

an e^m-th power residue of n is less than or equal to e^{-m}.

Proof. Let $n_i = p_i^{a_i}$ be a prime power of the factorization of n such that $e^m \mid \phi(n_i)$. Since n is odd, n_i possesses a primitive root. Let g be a primitive root of n_i. For an integer γ randomly selected from $\mathbb{Z}_n^*$, let $\mathrm{ind}_g\gamma$ denote the index of γ to the base g modulo n_i. Then γ is an e^m-th power residue of n_i if and only if the congruence $x^{e^m} \equiv v \pmod{n_i}$ has a solution, or equivalently, if and only if

$$g^{e^m \,\mathrm{ind}_g x - \mathrm{ind}_g \gamma} \equiv 1 \pmod{n_i},$$

which is equivalent to

$$e^m \mathrm{ind}_g x \equiv \mathrm{ind}_g \gamma \pmod{\phi(n_i)}.$$

Since $e^m \mid \phi(n_i)$, γ is an e^m-th power residue of n_i if and only if $e^m \mid \mathrm{ind}_g\gamma$.

Let $n_i' = n/n_i$, then n_i and n_i' are relatively prime. For any integer $\beta \in \mathbb{Z}_n^*$, it is clear that $\beta \mod n_i$ and $\beta \mod n_i'$ are integers of $\mathbb{Z}_{n_i}^*$ and $\mathbb{Z}_{n_i'}^*$, respectively. On the other hand, for two integers $\alpha_1 \in \mathbb{Z}_{n_i}^*$ and $\alpha_2 \in \mathbb{Z}_{n_i'}^*$, by the Chinese Remainder Theorem, there is an unique integer $\alpha \in \mathbb{Z}_n^*$, such that $\alpha \equiv \alpha_1 \pmod{n_i}$, and $\alpha \equiv \alpha_2 \pmod{n_i'}$. So, the number of integers $\alpha \in \mathbb{Z}_n^*$ which satisfy the congruence $\alpha \equiv \alpha_1 \pmod{n_i}$ is $\phi(n_i')$. If γ is randomly selected from $\mathbb{Z}_n^*$, then for any integer s, $0 \le s \le \phi(n_i) - 1$, we have

$$Pr(g^s = \gamma \mod n_i) = \phi(n_i')/\phi(n) = 1/\phi(n_i),$$

which implies that

$$Pr(\mathrm{ind}_g\gamma = s) = 1/\phi(n_i).$$

Hence,

$$\begin{aligned} Pr(e^m \mid \mathrm{ind}_g\gamma) &= \sum_{e^m \mid s,\, 0 \le s < \phi(n_i)} Pr(\mathrm{ind}_g\gamma = s) \\ &= \phi(n_i)e^{-m}/\phi(n_i) \\ &= e^{-m} \end{aligned}$$

which indicates that, for an integer γ randomly selected from $\mathbb{Z}_n^*$, the probability that γ is an e^m-th power residue of n_i is equal to e^{-m}. So, the probability that γ is an e^m-th power residue of n does not exceed e^{-m}. □

Theorem 2 demonstrates that, if there exits a prime-power $p_i^{a_i}$ of the factorization of n such that $e^m \mid \phi(p_i^{a_i})$, then for a random number $\gamma \in \mathbb{Z}_n^*$, the probability that Alice can decrypt γ repeatedly m times is less than or equal to e^{-m}. If the number u received from Alice satisfies the equation $E^m(u) = u^{e^m} = \gamma \mod n$, i.e., γ is an e^m-power residue of n, then Bob is ensured with probability greater than or equal to $1 - e^{-m}$ that, for every prime-power $p_i^{a_i}$ of the factorization of n, $e^m \nmid \phi(p_i^{a_i})$. Since $m = \lceil \log_e \varepsilon^{-1} \rceil$, $e^{-m} \le \varepsilon$. By Theorem 1, it is clear that the probability for an adversary to launch a successful e-residue attack against CEKEP is upper-bounded by ε

In the protocol CEKEP, the computational burden on Bob includes two modulo exponentiations, i.e., $u^{e^m} \bmod n$ and $(\lambda a^e)^{e^{m-1}} \bmod n$, where $m = \lceil \log_e \varepsilon^{-1} \rceil$. When $e < \varepsilon^{-1}$, each modulo exponentiation has an exponent consisting of $\mathcal{O}(\lceil \log_2 \varepsilon^{-1} \rceil)$ bits. The computation time for the two modulo exponentiations is $\mathcal{O}(2(\log_2 \varepsilon^{-1})(\log_2 n)^2)$. If $\varepsilon^{-1} \ll n$, then the computational load on Bob is greatly reduced in CEKEP in comparison with that in PEKEP (or in SNAPI). The parameter ε determines the computational load on Bob. It also determines the level of security against e-residue attacks. In practice, Bob can make a trade-off between the computational load and the security level offered by the protocol. When $\varepsilon = 2^{-80}$, for example, Bob needs to compute two modular exponentiation, each

having an exponent of about 80 bits. In this case, the computational load on Bob is lighter than that in a Diffie-Hellman based password-authenticated key exchange protocol. In the Diffie-Hellman based EKE variant, for example, Bob needs to compute two modular exponentiation, each having an exponent of at least 160 bits.

The computational load on Bob could be reduced further if Bob maintains a cache in the protocol CEKEP (or PEKEP). The cache stores a hashed version of Alice's public key (n, e) used in a previous sessions, that is, $H_1(n, e, A)$. Initially, the cache is empty. In the next run of CEKEP (PEKEP), Bob computes the hashed value of the public key received from Alice and checks if it is in the cache. If it is in the cache, Bob sets $m = 1$ ($m = 0$ for PEKEP) and then directly sends r_B, z, m to Alice (Bob does not send ϱ, m to Alice). Otherwise, Bob and Alice behave as usual (i.e. as specified in Fig. 1 or Fig. 2) and at the end of a success run, Bob updates the cache using Alice's new public key. In general, Alice would most likely use the same RSA key pairs in many sessions, although for perfect forward secrecy, Alice would need to choose a new key pair in each session. In such a circumstance, Bob may compute a single RSA encryption in a run of CEKEP (PEKEP).

6 Formal Security Analysis

In this section, we analyze the security of PEKEP and CEKEP within the formal model of security given in Section 3. Our analysis is based on the random-oracle model of Bellare and Rogaway [7]. In this model, a hash function is modeled as an oracle which outputs a random number for each new query. If the same query is asked twice, identical answers are returned by the oracle. In our analysis, we also assume the intractability of the RSA problem.

RSA Assumption: Let ℓ be the security parameter of RSA. Let key generator GE define a family of RSA functions, i.e., $(e, d, n) \leftarrow GE(1^\ell)$, where n is the product of two primes of the same size, $\gcd(e, \phi(n)) = 1$, and $ed \equiv 1 \pmod{\phi(n)}$. For any probabilistic polynomial-time algorithm $\mathcal{C}$, the following probability

$$Pr(x^e = c \mod n : (e, d, n) \leftarrow GE(1^\ell), c \in_R \{0,1\}^\ell, x \leftarrow \mathcal{C}(1^\ell, c, e, n))$$

is negligible.

Under the above assumptions, we have the following Theorem 3. The proof of this theorem is given in Appendix A.

Theorem 3 *Both protocols, PEKEP and CEKEP, are secure password-authenticated key exchange protocols under the RSA assumption and the random oracle model.*

We notice that the random oracle model in Theorem 3 is less desirable than a standard cryptographic assumption. To avoid the random oracle model, we could use the proof technique of [13], which require a public-key encryption scheme secure against chosen-ciphertext attacks. Unfortunately, the most commonly used RSA schemes (e.g. [6, 8]) which are secure against chosen-ciphertext attacks are also based on the random oracle model. Nevertheless, it is encouraging to see that efficient password-authenticated key exchange protocols with security proof in the random oracle model can be constructed without severe restriction on the public key of RSA.

7 Conclusion

In this paper, we investigate the design of RSA-based password-authenticated key exchange protocols that do not restrict the size of RSA public exponent. Based on number-theoretic techniques, we

develop a password enabled key exchange protocol (PEKEP) which allows using both large and small primes as RSA public exponents. We show that the protocol PEKEP is secure against e-residue attacks. We also provide a formal security analysis of PEKEP under the RSA assumption and the random oracle model. Based on PEKEP, we develop a computationally-efficient key exchange protocol to mitigate the burden on communication entities.

In both protocols PEKEP and CEKEP, Alice and Bob only need to establish a shared password beforehand; they do not need to establish other common parameters such as a prime number p and a generator of the cyclic group modulo p. This provides convenience both for the implementation and for the application of the two protocols. One the other hand, both PEKEP and CEKEP require that Alice and Bob have cleartext access to the shared password. When the two protocols are used for client-server communications, the server has cleartext access to all the passwords of its clients. This may raise concern on the database security of the server. To mitigate the risk of server compromise, we recommend that storage-security techniques [9, 14] be implemented in the server side.

References

[1] E. Bach and J. Shallit, *Algorithmic Number Theory*, vol. 1: *Efficient Algorithms*, MIT Press, 1997.

[2] S. M. Bellovin and M. Merritt, Encrypted key exchange: Password-based protocols secure against dictionary attacks, *Proc. of the IEEE Symposium on Research in Security and Privacy*, Oakland, May 1992, pp. 72-84.

[3] S. M. Bellovin and M. Merritt, Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise, *Proc. of the 1st ACM Conference on Computer and Communications Security*, ACM, November 1993, pp. 244-250.

[4] V. Boyko, P. MacKenzie, and S. Patel, Provably secure password authenticated key exchange using Diffie-Hellman, *Advances in Cryptology - EUROCRYPT 2000 Proceedings*, Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, 2000, pp. 156-171.

[5] M. Bellare, D. Pointcheval, and P. Rogaway, Authenticated key exchange secure against dictionary attack, *Advances in Cryptology - EUROCRYPT 2000 Proceedings*, Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, 2000, pp. 139-155.

[6] M. Bellare and P. Rogaway, Optimal asymmetric encryption, *Advances in Cryptology - EUROCRYPT '94 proceedings*, Lecture Notes in Computer Science, vol. 950, Springer-Verlag, 1995, pp. 92–111.

[7] M. Bellare and P. Rogaway, Entity Authentication and key distribution, *Advances in Cryptology - Crypto 93 Proceedings*, Lecture Notes in Computer Science Vol. 773, Springer-Verlag, 1994, pp. 22-26.

[8] D. Boneh, Simplified OAEP for the RSA and Rabin functions, *Advances in Cryptology - Crypto 2001 Proceedings*, Lecture Notes in Computer Science, Vol. 2139, Springer-Verlag, 2001, pp. 275-291.

[9] J. Chirillo and S. Blaul, *Storage Security: Protecting, SANs, NAS, and DAS*, John Wiley & Sons, 2002.

[10] R. Gennaro and Y. Lindell, A framework for password-based authenticated key exchange, *Advances in Cryptology - Eurocrypt 2003 Proceedings*, Lecture Notes in Computer Science Vol. 2656, Springer-Verlag, 2003, pp.524-542.

[11] O. Goldreich and Y. Lindell, Session-key generation using human passwords only, *Advances in Cryptology - Crypto 2001 Proceedings*, Lecture Notes in Computer Science Vol. 2139, Springer-Verlag, 2001, pp.408-432.

[12] L. Gong, Optimal authentication protocols resistant to password guessing attacks, *Proc. IEEE Computer Security Foundation Workshop*, June 1995, pp. 24-29

[13] S. Halevi and H. Krawczyk, Public-key cryptography and password protocols, *Proc. of the Fifth ACM Conference on Computer and Communications Security*, 1998, pp. 122-131,

[14] J. Hughes and J. Cole, Security in storage, *IEEE Computer*, January 2003, pp. 124-125.

[15] D. Jablon, Strong password-only authenticated key exchange, *Computer Communication Review, ACM SIGCOMM*, vol. 26, no. 5, 1996, pp. 5-26.

[16] D. Jablon, http://www.integritysciences.com.

[17] J. Katz, R. Ostrovsky, and M. Yung, Efficient password-authenticated key exchange using human-memorable passwords, *Advances in Cryptology – Eurocrypt'2001 Proceedings*, Lecture Notes in Computer Science, Vol. 2045, Springer-Verlag, 2001.

[18] K. Kobara and H. Imai, Pretty-simple password-authenticated key-exchange under standard assumptions, *IEICE Trans.*, vol. E85-A, no. 10, 2002, pp. 2229-2237.

[19] T. Kwon, Authentication and key agreement via memorable passwords, *Proc. Network and Distributed System Security Symposium*, February 7-9, 2001.

[20] M. Lamos, L. Gong, J. Saltzer, and R. Needham, Reducing risks from poorly chosen keys, *Proc. of the 12th ACM Symposium on Operating System Principles, ACM Operating Systems Review*, 1989, pp. 14-18.

[21] S. Lucks, Open key exchange: How to defeat dictionary attacks without encrypting public keys, *Proc. Security Protocol Workshop*, Lecture Notes in Computer Science, Vol. 1361, Springer-Verlag, 1997, pp. 79-90.

[22] P. MacKenzie, S. Patel, and R. Swaminathan, Password-authenticated key exchange based on RSA, *Advances in Cryptology—ASIACRYPT 2000 Proceedings*, Lecture Notes in Computer Science, vol. 1976, Springer-Verlag, 2000, pp. 599–613.

[23] A. Menezes, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[24] S. Patel, Number theoretic attacks on secure password schemes, *Proc. IEEE Symposium on Security and Privacy*, Oakland, California, May 5-7, 1997.

[25] K. H. Rosen, *Elementary Number Theory and Its Applications*, 4th ed., Addison Wesley Longman, 2000.

[26] T. Wu, The secure remote password protocol , *Proc. Network and Distributed System Security Symposium*, San Diego, March 1998, pp. 97-111.

[27] T. Wu, A real-world analysis of Kerberos password security, *Proc. Network and Distributed System Security Symposium*, February 3-5, 1999.

[28] F. Zhu, D. Wong, A. Chan, and R. Ye, RSA-based password authenticated key exchange for imbalanced wireless networks, *Proc. Information Security Conference 2003 (ISC'02)*, Lecture Notes in Computer Science, vol. 2433, Springer-Verlag, 2002, pp.150-161.

A Proof of Theorem 3

It is easy to verify that the protocol PEKEP satisfies the first condition of Definition 1. To prove that the protocol PEKEP also satisfies the second condition of Definition 1, we first prove the following Lemma 1:

Lemma 1 *Let $\mathcal{A}$ be a polynomial-time adversary who makes v impersonation attempts in attacking the protocol PEKEP. Let $\pi_1, \pi_2, \ldots, \pi_v$ denote her guesses of the password w of A and B in the v impersonation attempts. Let E_v denote the event that one of her guesses, say π_i, is a correct guess and let $\neg E_v$ denote that event that $\pi_1, \pi_2, \ldots, \pi_v$ are incorrect guesses. Then, under the condition that $\neg E_v$ is true, the adversary's success probability in attacking the protocol PEKEP is*

$$Pr(\mathsf{Succ}|_{\neg E_v}) = 1/2 + \zeta,$$

where ζ is negligible.

Proof. As described in Section 3, the adversary $\mathcal{A}$ always uses an impersonated instance to launch an impersonation attack. The adversary may guess the password of A and B at the beginning of the attack or in the middle of the attack. Assume that the adversary $\mathcal{A}$ makes a Test query on a *fresh* instance, which is either Π_A^i or Π_B^j, and succeeds with probability $Pr(\mathsf{Succ})$. To prove that $Pr(\mathsf{Succ}|_{\neg E_v}) - 1/2$ is negligible, we consider the following two cases:

Case 1: Test query is called on Π_A^i. Assume that the instance Π_A^i sent out r_A, n, e, A in the first flow and received r_B and z in the second flow. Also assume that the instance Π_A^i queried the oracle H on the input w, r_A, r_B, A, B, n, e and received an answer α. When $\gcd(\alpha, n) = 1$, the instance computed $b = D(\alpha^{-1}D^m(z))$, where $m = \lfloor \log_e n \rfloor$. Then the instance Π_A^i queried the oracle H_1 on the input b, r_A, r_B, A, B, n, e and obtained the answer μ. After receiving η in the last flow, the instance Π_A^i accepted and generated the session key sk_B^j by querying the oracle H_3 on the input b, r_A, r_B, A, B, n, e. Without loss of generality, we assume that random numbers generated by Π_A^i and by random oracles H, H_1, H_2, H_3 never repeat. Let Auth denote the event that r_B, z and η were sent by an instance which was not impersonated by $\mathcal{A}$. In the following, we show that the probability $Pr(\neg\mathsf{Auth}|_{\neg E_v})$ is negligible.

(a) Assume that r_B, z were sent by an instance Π_B^j which was impersonated by $\mathcal{A}$. After receiving r_A, n, e, A from Π_A^i, the impersonated instance Π_B^j generated two random numbers $a \in \mathbb{Z}_n^*$ and $r_B \in \{0,1\}^k$. Let α' denote the answer of the oracle H on the input $\pi, r_A, r_B, A, B, n, e$, where π is the adversary's guess of the password of A. Since (n, e) is valid RSA public key, the probability that α' is not relatively prime to n is negligible. Without loss of generality, assume that $\gcd(\alpha', n) = 1$, then $z = E^m(\alpha' E(a))$. Under the condition that $\neg E_v$ is true, we have $\alpha' \neq \alpha$, which implies that

$$b = D(\alpha^{-1}D^m(z)) \neq a.$$

Hence, the message η was not sent by the impersonated instance Π_B^j. Next, we show that, except with negligible probability, η could not be sent by $\mathcal{A}$. Due to the randomness assumption of H_2, the adversary needs to know b in order to generate η. Assume that the adversary can recover the integer

b from z and μ with probability p_b. Then, for any integers λ and z randomly selected from $\mathbb{Z}_n^*$, the adversary can obtain the solution of the congruence

$$(\lambda x^e)^{e^m} = z \mod n \tag{6}$$

with probability p_b. If p_b is non-negligible, we turn the adversary $\mathcal{A}$ into an efficient algorithm $\mathcal{C}$ for the RSA problem as follows: given $c \in_R \mathbb{Z}_n^*$, algorithm $\mathcal{C}$ selects $\lambda \in_R \mathbb{Z}_n^*$ and computes $z = (\lambda c)^{e^m}$, then algorithm $\mathcal{C}$ runs the adversary $\mathcal{A}$ to obtain the solution of (6). It is clear that if β is the solution of (6), then $\beta^e = c \bmod n$, i.e., β is the plaintext of c. Hence, under the RSA assumption, the probability for $\mathcal{A}$ to recover b is negligible. So, under the condition $\neg E_v$, the probability that r_B, z and η were sent by $\mathcal{A}$ or by an impersonated instance is negligible.

(b) Assume that r_B, z and η were sent by an instance Π_B^j which was not impersonated by $\mathcal{A}$. Then Π_B^j is partnered with Π_A^i. Under the assumption that random numbers generated by entity instances and by random oracles never repeat, it is clear that Π_B^j is the only instance partnered with Π_A^i. Thus, the session key sk_A^i could not be held by any instance other than Π_A^i and Π_B^j. Due to the randomness assumption of H_3, the session key sk_A^i is just a random session key for anyone without knowing the integer b shared between A and B. From the above analysis, if the adversary $\mathcal{A}$ can recover b with non-negligible probability, then we can turn $\mathcal{A}$ into an efficient algorithm $\mathcal{C}$ for the RSA problem. Thus, under the RSA assumption, the probability for $\mathcal{A}$ to recover b is negligible.

(c) Let Secc denote the event that $\mathcal{A}$ does not know the session key sk_A^i of Π_A^i. The analysis in (a) and (b) shows that both $Pr(\neg\mathsf{Auth}|_{\neg E_v})$ and $Pr(\neg\mathsf{Secc}|_{\mathsf{Auth}})$ are negligible. Hence, we have

$$\begin{aligned}
Pr(\mathsf{Succ}|_{\neg E_v}) &= Pr(\mathsf{Succ}|_{\neg\mathsf{Auth}})Pr(\neg\mathsf{Auth}|_{\neg E_v}) + Pr(\mathsf{Succ}|_{\mathsf{Auth}})Pr(\mathsf{Auth}|_{\neg E_v}) \\
&\leq Pr(\neg\mathsf{Auth}|_{\neg E_v}) + Pr(\mathsf{Succ}|_{\mathsf{Auth}}) \\
&= Pr(\neg\mathsf{Auth}|_{\neg E_v}) + Pr(\mathsf{Succ}|_{\mathsf{Secc}})Pr(\mathsf{Secc}|_{\mathsf{Auth}}) + Pr(\mathsf{Succ}|_{\neg\mathsf{Secc}})Pr(\neg\mathsf{Secc}|_{\mathsf{Auth}}) \\
&\leq Pr(\neg\mathsf{Auth}|_{\neg E_v}) + Pr(\mathsf{Succ}|_{\mathsf{Secc}})(1 - Pr(\neg\mathsf{Secc}|_{\mathsf{Auth}})) + Pr(\neg\mathsf{Secc}|_{\mathsf{Auth}})
\end{aligned}$$

Note that the success probability of $\mathcal{A}$ is equal to $1/2$ if $\mathcal{A}$ does not know the session key sk_A^i of Π_A^i, that is, $Pr(\mathsf{Succ}|_{\mathsf{Secc}}) = 1/2$. Therefor, it follows that

$$Pr(\mathsf{Succ}|_{\neg E_v}) \leq \frac{1}{2} + Pr(\neg\mathsf{Auth}|_{\neg E_v}) + \frac{1}{2}Pr(\neg\mathsf{Secc}|_{\mathsf{Auth}}),$$

which indicates that $Pr(\mathsf{Succ}|_{\neg E_v}) - 1/2$ is negligible.

Case 2: Test query is called on Π_B^j. Assume that the instance Π_B^j received r_A, n, e, A in the first flow and sent out r_B and z in the second flow, where $z = (\lambda a^e)^{e^m} \bmod n$, $a \in_R \mathbb{Z}_n^*$. The integer λ is equal to $\alpha = H(w, r_A, r_B, A, B, n, e)$ if $\gcd(\alpha, n) = 1$, and else, λ is a random number of $\mathbb{Z}_n^*$. The instance Π_B^j accepted after receiving μ, which is equal to the answer of the oracle H_1 on the input a, r_A, r_B, A, B, n, e. As in Case 1, we first show that, under the condition $\neg E_v$, the probability that r_A, n, e and μ were sent by $\mathcal{A}$ or by an impersonated instance is negligible.

(a) Assume that r_A, n, e was sent by an instance Π_A^i which was impersonated by $\mathcal{A}$. The adversary selected a password $\pi \in \mathcal{D}$ and a pair of odd integers n and e for the impersonated instance Π_A^i. Note that (n, e) is not necessarily a valid RSA public key. Let α' denote the answer of the oracle H on the input $\pi, r_A, r_B, A, B, n, e$. Then, under the condition that E_v is false, we have $\alpha \neq \alpha'$. If $\gcd(\alpha', n) \neq 1$, the impersonated instance Π_A^i selects a random number $b \in \mathbb{Z}$ for the computation of μ. In this scenario, we have

$$Pr(b = a) = 1/\phi(n).$$

For all integers $n \geq 5$, it is known (see [23]) that

$$\phi(n) > n/(6 \ln\ln n).$$

Thus, the probability that μ was sent by Π_A^i is negligible if $\gcd(\alpha', n) \neq 1$. Now, let us assume that α' is relatively prime to n. By Theorem 1, the congruence

$$(\alpha' x^e)^{e^m} = z \mod n \tag{7}$$

has solutions in $\mathbb{Z}_n^*$. On the other hand, since z is an e^m-th power residue of n, for any integer $b \in \mathbb{Z}_n^*$, there exists an $\lambda \in \mathbb{Z}_n^*$ such that b is a solution of the congruence

$$(\lambda x^e)^{e^m} = z \mod n.$$

Hence, for a random number α' not equal to α, the probability that a is a solution of (7) is equal to $1/\phi(n)$. Therefore, under the condition $\neg E_v$, the probability that r_A, n, e and μ were sent by the impersonated instance Π_A^i is negligible.

(b) Next, assume that r_A, n, e and μ were sent by an instance which was not impersonated by $\mathcal{A}$. As shown in Case 1(b), the probability that $\mathcal{A}$ can recover the session key sk_B^j of Π_B^j is negligible. Following the analysis in Case 1(c), it can be proved that $Pr(\mathsf{Succ}|_{\neg E_v}) - 1/2$ is also negligible in Case 2. □

Proof of Theorem 3. Now, we prove that the protocol PEKEP satisfies the second condition of Definition 1. Let us fix a polynomial-time adversary $\mathcal{A}$ who makes v impersonation attempts in attacking the protocol PEKEP. Let $\pi_1, \pi_2, \ldots, \pi_v$ denote her guesses of the password of A and B in the v impersonation attempts. Let E_v denote the event that one of her guesses, say π_i, is a correct guess. Under the condition that E_v is true, it is clear that the adversary's success probability $Pr(\mathsf{Succ})$ in attacking the protocol PEKEP would be 1, i.e., $Pr(\mathsf{Succ}|_{E_v}) = 1$. Let $\neg E_v$ denote that event that E_v is false, i.e., $\pi_1, \pi_2, \ldots, \pi_v$ are incorrect guesses. By Lemma 1, $\zeta = Pr(\mathsf{Succ}|_{\neg E_v}) - 1/2$ is negligible. Hence, we have

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{ake} &= 2Pr(\mathsf{Succ}) - 1 \\
&= 2Pr(\mathsf{Succ}|_{E_v})Pr(E_v) + 2Pr(\mathsf{Succ}|_{\neg E_v})Pr(\neg E_v) - 1 \\
&\leq \frac{2v}{|\mathcal{D}|} + 2(0.5 + \zeta)(1 - \frac{v}{|\mathcal{D}|}) - 1 \\
&= \frac{v}{|\mathcal{D}|} + \epsilon
\end{aligned}$$

where $\epsilon = 2\zeta(1 - v/|\mathcal{D}|)$. So, PEKEP is a secure password-authenticated key exchange protocol.

Following the security analysis for the protocol PEKEP, it can be proved that the protocol CEKEP also satisfies the two conditions of Definition 1 and thus is a secure password-authenticated key exchange protocol. □