More Efficient Password Authenticated Key Exchange Based on RSA

Duncan S. Wong[1], Agnes H. Chan[2], and Feng Zhu[2]

[1] Department of Computer Science
City University of Hong Kong
Hong Kong, China
duncan@cityu.edu.hk

[2] College of Computer Science
Northeastern University
Boston, MA 02115, U.S.A.
{ahchan,zhufeng}@ccs.neu.edu

Abstract. In [17], Zhu, et al. proposed a RSA-based password authenticated key exchange scheme which supports short RSA public exponents. The scheme is the most efficient one among all the RSA-based schemes currently proposed when implemented on low-power asymmetric wireless networks. We observe that its performance can further be improved by proposing two modifications. The first modification shortens the size of the message sent from the server to the client. The second modification dramatically reduces the size of the message sent from the client to the server and therefore can be used to reduce the power consumption of the client for wireless communications in a significant way. We also generalize our modified schemes and formalize the security requirements of all underlying primitives that the generic scheme is constituted. A new primitive called password-keyed permutation family is introduced. We show that the security of our password-keyed permutation family is computationally equivalent to the RSA Problem in the random oracle model.

Keywords: Password Authentication, Key Exchange, Secure Wireless Communications

1 Introduction

We investigate methods of providing efficient password authenticated key exchange (PAKE) for wireless communications between a low-power client and a powerful server. The objective of a password authenticated key exchange scheme is the same as a conventional authenticated key exchange scheme [5]: after two communicating parties successfully executing the scheme, each of them should have certain assurance that it knows each other's true identity (*authentication*), and it shares a new and random session key only with each other and the key is derived from contributions of both parties (*key exchange*). Unlike a

T. Johansson and S. Maitra (Eds.): INDOCRYPT 2003, LNCS 2904, pp. 375–387, 2003.

cryptographic-key authenticated key exchange scheme, the two communicating parties do not have any pre-shared cryptographic symmetric key, certificate or support from a trusted third party. Instead they only share a password. The major difficulty in designing a secure password-based protocol is due to the concern of implicated off-line dictionary attacks against a small password space [3]. A password, a passphrase, or a PIN (Personal Identification Number) generally needs to be easy to remember. Usually it has significantly less randomness than its length suggested or is simply very short in length. In our study, the password space is considered to be so small that an adversary can enumerate it efficiently.

We focus our attention on designing a password authenticated key exchange scheme for wireless communications between a low-power client and a powerful server. A powerful server has comparable computational power and memory capacity to a current desktop machine while a low-power client is as resource constrained as a smart card, a wearable device in a wireless PAN (Personal Area Network), a low-end PDA (Personal Digital Assistant) or a cellular phone. In addition, we consider the client to be mostly battery-powered with inferior communication capability. A typical cellular network in which a mobile unit communicating with a base station, a Bluetooth-based PAN in which a watch-size PDA communicating with a laptop, and a disposable sensor exchanging information with a more capable tandem device in an ad hoc sensor network are some typical examples of our target applications. Therefore, the objectives of our scheme design include optimizing the computational complexity especially at the client side, minimizing memory footprint and reducing the size of the messages exchanged between the two communicating parties. Besides the conventional techniques for pursuing these objectives, such as precomputation and caching, we also explore the fact of receiving radio signal consumes much less power than transmitting in scheme design.

1.1 Related Work

There were many password authenticated key exchange (PAKE) schemes proposed in the last decade, especially recently [3, 9, 16, 4, 13, 11, 6, 8][3]. Most of these schemes are based on Diffie-Hellman key exchange and perform large modular exponentiation operations which may take a long time to compute on a low-power device.

In [3], Bellovin and Merritt investigated the feasibility of using RSA [14] to construct a PAKE. If the RSA public exponent is short, the encryption operation can be done efficiently. However, they also pointed out that an e-residue attack may be feasible if the receiving party has no way to verify whether a RSA public exponent is fraudulent or not, that is, to check if the public exponent is relatively prime to $\phi(n)$ without knowing the factorization of a RSA modulus n. To thwart this attack, the authors considered an interactive protocol for validating the public exponent. However, the protocol was found to be insecure [17].

[3] Jablon maintains an updated list at http://www.integritysciences.com/links.html

Two other RSA based schemes were proposed later in [12, 13]. The first one was later found to be insecure, while the second one has to use a large prime for the public exponent. This defeats the purpose of using RSA for low-power clients in our target applications because the computational complexity of performing a RSA encryption is no less than that of carrying out a modular exponentiation in a Diffie-Hellman key exchange based protocol.

In [17], Zhu, et al. modified the interactive protocol in [3] and proposed a scheme which supports short RSA public exponents. The scheme is the most efficient one among all the RSA-based schemes currently proposed when implemented on low-power asymmetric wireless networks. For applications requiring moderate level of security, that is, if 512-bit RSA is used, the scheme takes less than 2.5 seconds of pure computation on a 16MHz Palm V and about 1 second for data transmission if the throughput of a network is only 8kbps. The computation time can be improved to 300 msec and the transmission time can also be reduced to 300 msec if caching is allowed. For applications where 1024-bit RSA is used, the total computation time of the client is estimated to be about 9 seconds according to the performance measurements of various cryptographic operations reported in [15]. It can be reduced dramatically to less than half second if the server's public key is cached at the client side. Since a client is usually communicating with an essentially fixed set of servers in most cases, the first 9-second protocol run can be considered as a one-time setup phase in a system.

1.2 Our Contributions

In [17], the most time consuming part of the scheme is to check the validity of the server's RSA public exponent. The checking mechanism is a two-round interactive protocol which requires 1KB of data sent from the client to the server and another 200B of data sent from the server to the client, if 1024-bit RSA is used with the same configurations of all other parameters specified in [17]. In this paper, we propose two modifications of the interactive protocol for improving its efficiency. The first modification reduces the size of the message sent from the server to the client. It is a straightforward modification but reduces the transmission time, memory footprint and computation complexity all at the same time. The second modification reduces the size of the message sent from the client to the server significantly and hence entails much less power consumption for the low-power client in wireless communications than the original protocol in [17] as receiving radio signal requires much less power than transmitting some. Memory footprint at the client side is also minimized and other optimization techniques such as precomputation and caching are preserved.

On the security analysis, we generalize our modified schemes to a generic one and formalize the security requirements of all underlying primitives that the generic scheme is constituted. In the formalization, we introduce a new primitive called password-keyed permutation family to capture the features of the password related operations in our schemes. We also show that the security of our password-keyed permutation family is computationally equivalent to the RSA Problem in the random oracle model [2].

The rest of the paper is organized as follows. In Sec. 2, the scheme proposed in [17] is reviewed. This is followed by the description of our two modifications in Sec. 3. In Sec. 4, we formalize our modifications by describing a generic scheme and specifying the security requirements of its underlying primitives. We conclude the paper in Sec. 5.

2 A RSA-based PAKE [17] — RSA-PAKE1

In [17], Zhu, et al. proposed a RSA-based password authenticated key exchange (PAKE) scheme which supports short RSA public exponents. It is refined later in [1] to eliminate potential vulnerabilities of using symmetric encryption function. In the following, we review the refined scheme with some modifications so that the final session key is generated from the contributions of both communicating parties. We call the new scheme the RSA-PAKE1.

Define some integer k as a system-wide security parameter. Let two finite-length strings $A, B \in \{0,1\}^*$ denote a powerful server and a low-power client, respectively. Let (n, e) be the RSA public key of A where n is the RSA modulus and e is the public exponent. Suppose A and B share a password $pw \in PW$ where PW denotes a password space in which the password is chosen according to certain probability distribution. Let $G_1, G_2, G_3, G_4, G_5 : \{0,1\}^* \to \{0,1\}^k$ be distinct and independent cryptographically strong hash functions. The protocol proceeds as follows.

1. A selects $r_A \in_R \{0,1\}^k$ and sends $((n,e),\ r_A)$ to B.
2. B checks if (n,e) is a valid public key using an Interactive Protocol (read Sec. 3 for details). It then randomly picks $r_B \in_R \{0,1\}^k$ and $s_B \in_R \mathbb{Z}_n^*$, and computes $\pi = T(pw, A, B, r_A, r_B)$ where $T : \{0,1\}^* \to \mathbb{Z}_n^*$ is a distinct cryptographic hash function. It sends r_B and $z = s_B^e \cdot \pi \bmod n$ to A.
3. A computes π accordingly and obtains s_B from z. It then computes $K = G_1(s_B)$, $c_B = G_3(s_B)$ and $\sigma = G_4(c_A, c_B, A, B)$, randomly picks $c_A \in_R \{0,1\}^k$, and sends $(K \oplus c_A,\ G_2(K, c_A, A, B))$ to B.
4. B computes K and c_B from s_B accordingly. It reveals c_A from the first part of the incoming message and checks if the second part of the incoming message is $G_2(K, c_A, A, B)$. It then computes the session key σ accordingly and sends $G_5(\sigma)$ back to A.
5. A finally checks if the incoming message is $G_5(\sigma)$.

Here we consider the RSA to be a trapdoor permutation over $\mathbb{Z}_n^*$. The operation $s_B^e \cdot \pi \bmod n$ can also be considered as a permutation of $s_B^e \bmod n$ over $\mathbb{Z}_n^*$ where $\pi \in \mathbb{Z}_n^*$. Observing that RSA is a trapdoor permutation also over $\mathbb{Z}_n$, we can use the following two permutation methods to compute z as well. Method 1: Let T be defined as $T : \{0,1\}^* \to \{0,1\}^{|n|-1}$. Compute z as $(s_B^e \bmod n) \oplus \pi$ for all $(s_B^e \bmod n) \in \{0,1\}^{|n|-1}$. Since not all randomly chosen elements in $\mathbb{Z}_n$ after being encrypted are $|n| - 1$ bits long, this method is probabilistic. Method 2: Compute z as $s_B^e + \pi \bmod n$ for all $s_B \in \mathbb{Z}_n$. For simplicity, we skip detail discussions of these two methods and focus our attention on improving the efficiency of the Interactive Protocol for checking the validity of (n, e) in Step 2 above.

3 Improving the Interactive Protocol

In RSA-PAKE1 described above, there is an Interactive Protocol for checking the validity of public keys. To ensure that the RSA cryptosystem works correctly, the public exponent e has to be relatively prime to $\phi(n)$ where n is a RSA modulus. If e is a fraudulent value, an active attacker may be able to launch various e-residue attacks [3, 12, 13].

The idea of using an interactive protocol to detect fraudulent values of e was first discussed in [3]. That is, after a verifier receives (n, e) from a prover, the verifier checks if n and e are odd. Then it picks N (say $N = 10$) integers $m_i \in_R \mathbb{Z}_n^*$, $1 \leq i \leq N$ and sends $\{c_i \equiv m_i^e \pmod{n}\}_{1 \leq i \leq N}$ to the prover. The prover computes the e-th root of each c_i as m_i' and sends $\{m_i'\}_{1 \leq i \leq N}$ back. Correct replies indicate that e has the proper form. In [17], Zhu, et al. found that this preliminary version is insecure. It allows an impersonator of B to test multiple trial passwords in one run of the Interactive Protocol with A. To prevent the attack, RSA-PAKE1 uses the following variant of the Interactive Protocol.

Let $h : \{0,1\}^* \rightarrow \{0,1\}^k$ be a collision-free hash function. After the verifier sends out $\{c_i\}_{1 \leq i \leq N}$, it stores $\{h(m_i)\}_{1 \leq i \leq N}$ instead of $\{m_i\}_{1 \leq i \leq N}$ in its memory. Similarly the prover sends back $\{h(m_i')\}_{1 \leq i \leq N}$. Due to the collision-free property of h, it is negligible to have $h(m) = h(m')$ for $m \neq m'$. The checking mechanism of the interactive protocol is retained and the attack against the preliminary version is prevented. In addition, this also reduces the size of the reply in the Interactive Protocol and reduces the memory footprint of the verifier.

In the following, we describe two modifications. The first one shortens the size of the message sent from the server to the client, while the second one dramatically reduces the size of the message sent from the client to the server and hence saves much more power than the first one for the low-power client in wireless communications as transmitting radio signal requires much more power than receiving some.

3.1 Modification 1

After the verifier sends $\{c_i\}_{1 \leq i \leq N}$ out, it stores $h(m_1, m_2, \cdots, m_N)$ in its memory. Similarly the prover sends $h(m_1', m_2', \cdots, m_N')$ back. This simple modification reduces the number of hash values to be stored and transferred from N to only one. It also reduces the hash operations at the client side from N times to one single hash operation.

In wireless communications, more power can be saved if the amount of data transmitted is reduced. In Modification 1, we reduce the amount of data needs to be received by the verifier (that is, the low-power client) while the amount of data sent by it remains the same. In Modification 2 below, we reduce the amount of data sent by the low-power client dramatically and therefore significantly reduce the power consumption of it.

3.2 Modification 2

For some odd integers n and $e \geq 3$, let eRES $= \{y = x^e \bmod n \ : \ x \in \mathbb{Z}_n^*\}$ be the e-residue set. If $(e, \phi(n)) \neq 1$, then $|\text{eRES}| \leq \phi(n)/3$ since each e-residue has at least 3 e-th roots. When N elements are randomly picked in $\mathbb{Z}_n^*$, the chance of having all of them in eRES is at most 3^{-N}. Base on this fact, we modify the Interactive Protocol as follows.

Define a random function H such that on inputs N, n, e, A, B, and two k-bit binary strings r_1 and r_2, the function generates a sequence of N random elements in $\mathbb{Z}_n^*$. This is denoted as $(c_1, c_2, \cdots, c_N) \leftarrow H(N, n, A, B, r_1, r_2)$. The prover (A) picks $r_1 \in_R \{0,1\}^k$ and sends (n, e, r_1) to the verifier (B). The verifier then replies with $r_2 \in_R \{0,1\}^k$. The prover computes $(c_1, c_2, \cdots, c_N) \leftarrow H(N, n, e, A, B, r_1, r_2)$ and sends $\{m_i = c_i^{1/e} \bmod n\}_{1 \leq i \leq N}$ to the verifier. By using H, the verifier generates $\{c_i\}_{1 \leq i \leq N}$ accordingly and checks if $c_i \equiv m_i^e \pmod n$ for all $1 \leq i \leq N$. The verifier accepts if all the checks are passed. By replacing the original interactive protocol of RSA-PAKE1 with Modification 2, we obtain a scheme shown in Fig. 1. We call it the RSA-PAKE2.

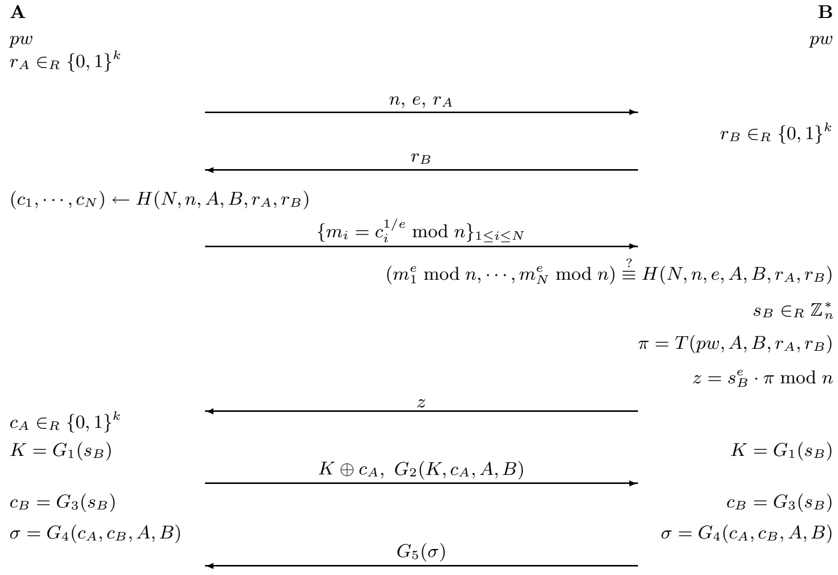
A
B
pw
pw
$r_A \in_R \{0,1\}^k$
$n,\ e,\ r_A$
$r_B \in_R \{0,1\}^k$
r_B
$(c_1, \cdots, c_N) \leftarrow H(N, n, A, B, r_A, r_B)$
$\{m_i = c_i^{1/e} \bmod n\}_{1 \leq i \leq N}$
$(m_1^e \bmod n, \cdots, m_N^e \bmod n) \overset{?}{\equiv} H(N, n, e, A, B, r_A, r_B)$
$s_B \in_R \mathbb{Z}_n^*$
$\pi = T(pw, A, B, r_A, r_B)$
$z = s_B^e \cdot \pi \bmod n$
z
$c_A \in_R \{0,1\}^k$
$K = G_1(s_B)$
$K = G_1(s_B)$
$K \oplus c_A,\ G_2(K, c_A, A, B)$
$c_B = G_3(s_B)$
$c_B = G_3(s_B)$
$\sigma = G_4(c_A, c_B, A, B)$
$\sigma = G_4(c_A, c_B, A, B)$
$G_5(\sigma)$

Fig. 1. RSA-PAKE2

In this modification, all the N random numbers are generated dynamically using the random function H. Hence the low-power client B does not need to store any of them in the memory and the memory footprint is further minimized when compared with Modification 1.

On the computation complexity, the number of modular multiplications carried out in RSA-PAKE2 is no more than that in RSA-PAKE1 as well as in Modification 1. Hence the computation time is estimated to be similar to that of the previous variants. That is, it takes 2.5 seconds to compute on a 16MHz Palm V and reduces to 300 msec if the server's public key is cached for subsequent protocol runs after the first protocol run when $N = 10$ and 512-bit RSA is used. For stringent security requirement when 1024-bit RSA is used, the first run of the protocol requires 9 second of pure computation but all the subsequent runs with the same server can be reduced to less than half second.

On the power consumption, the low-power client receives N random numbers rather than sending N numbers as required in RSA-PAKE1 and Modification 1. In addition, only one k-bit long random number is sent by the client in the Interactive Protocol of RSA-PAKE2. This reduces the transmission time of the Interactive Protocol over a 8kbps network by 180 msec for $k = 160$. In addition, since receiving data consumes much less energy than sending data in wireless communications, for 1024-bit RSA, the client of RSA-PAKE2 spends only 20 msec in sending data in the Interactive Protocol, while the client of RSA-PAKE1 spends 1.28 seconds in sending data. RSA-PAKE2's approach of having the low-power client dramatically reduce the message sent with the increase of message received can help the client to save power in a very significant way.

4 Formalization and the Generic Scheme

In this section, we generalize RSA-PAKE2 to a generic scheme and formalize the security requirements of all underlying primitives that the generic scheme is constituted. In the formalization, we introduce a new primitive called password-keyed permutation family to capture the features of the password related operations of RSA-PAKE1 (and the two modifications). We also show that these password related operations satisfy the security requirements of the password-keyed permutation family of the generic scheme if and only if the RSA Problem is hard.

4.1 The Public Key Encryption Function

From the conjectures that the RSA problem is hard and the RSA encryption primitive is a trapdoor one-way permutation over $\mathbb{Z}_n^*$, we formalize the security requirement of the public key encryption function defined by a public key $PK \in PubK(k)$ in the generic scheme to be a trapdoor one-way permutation, given by $\mathcal{E}_{PK} : \mathcal{S}(PK) \rightarrow \mathcal{S}(PK)$ where $\mathcal{S}(PK)$ is the set of elements over which the trapdoor permutation is defined. $PubK(k)$ denotes the set of public keys with respect to k. In general, we can relax the requirement to allow any trapdoor one-way function as the encryption function.

4.2 The Password-Keyed Permutation Family

In Fig. 1, z is computed as the encryption of a random element in $\mathbb{Z}_n^*$ followed by a modular multiplication with π where $\pi \in \mathbb{Z}_n^*$ is a function of the password pw with some nonces r_A and r_B and identification information. Similar to the definition of the public key encryption function above, we can also consider the modular multiplication as a permutation over the same set of elements as the public key encryption function. Define $T : \{0,1\}^* \rightarrow PwdK(k)$ to be a distinct and independent cryptographic hash function. We assume that the size of $PwdK(k)$ is at least 2^k. Let $\mathcal{P}^{PK} : PwdK(k) \times \mathcal{S}(PK) \rightarrow \mathcal{S}(PK)$ be a collection of permutations for every $PK \in PubK(k)$. For simplicity, we usually omit the superscript notation of the public key on $\mathcal{P}$. We call $\mathcal{P}$ a *password-keyed permutation family.* For every $\pi \in PwdK(k)$, we define a permutation $\mathcal{P}_\pi : \mathcal{S}(PK) \rightarrow \mathcal{S}(PK)$ by $\mathcal{P}_\pi(x) = \mathcal{P}(\pi, x)$. From these definitions, z is then computed in the generic scheme as $z = \mathcal{P}_\pi(\mathcal{E}_{PK}(s_B))$ where $s_B \in_R \mathcal{S}(PK)$.

We now discuss the security requirements of $\mathcal{P}$. The first requirement of $\mathcal{P}$ is *distinctness.* This means for every pair $(\pi_1, \pi_2) \in PwdK(k) \times PwdK(k)$, $\pi_1 \neq \pi_2$, and for any $y \in \mathcal{S}(PK)$, $\Pr[\mathcal{P}_{\pi_1}(y) = \mathcal{P}_{\pi_2}(y)] \leq \epsilon(k)$ where ϵ is some negligible function. It is not difficult to see that the purpose of having $\mathcal{P}$ be distinct is to prevent disturbing the probability distribution of picking pw from PW, which may provide an adversary with greater advantage in guessing the password.

Besides having $\mathcal{P}$ be distinct, we also require $\mathcal{P}$ to satisfy the following security requirement.

Definition 1. *Given a trapdoor one-way permutation $f : Dom(k) \rightarrow Dom(k)$, a password space PW, and a hash function $T : \{0,1\}^* \rightarrow PwdK(k)$ behaves like a random oracle, a distinct password-keyed permutation family $\mathcal{P}^f : PwdK(k) \times Dom(k) \rightarrow Dom(k)$ is secure if for every probabilistic polynomial-time algorithm E^T and for all sufficiently large k,*

$$\begin{aligned}\Pr[\ & E^T(1^k, A, B, PW, f, r_A) \rightarrow (z, r_B, x_1, x_2, \pi_1, \pi_2)\ :\ r_B \in \{0,1\}^k, \\ & z, x_1, x_2 \in Dom(k),\ \pi_1, \pi_2 \in \mathit{\Gamma}_{A,B,r_A,r_B},\ z = \mathcal{P}^f(\pi_1, f(x_1)) = \mathcal{P}^f(\pi_2, f(x_2))\ ] \\ & \leq \epsilon(l)\end{aligned}$$

for all $r_A \in \{0,1\}^k$, $A, B \in \{0,1\}^*$ and for some negligible function ϵ where

$$\mathit{\Gamma}_{A,B,r_A,r_B} = \{\ T(pw, A, B, r_A, r_B)\ : pw \in PW\ \}.$$

It means that an attacker should not be able to compute more than one pair of (π, x) such that $\mathcal{P}^f(\pi, f(x))$ produces the same value of z.

To understand the reason of specifying this security requirement, we consider a run of the generic scheme in which an adversary E is impersonating B. Suppose that after receiving PK and r_A from A, E has non-negligible success rate of constructing (z, r_B) and obtaining (at least) two values x_1, x_2 and corresponding $\pi_1, \pi_2 \in \mathit{\Gamma}_{A,B,r_A,r_B}$ such that $z = \mathcal{P}^f_{\pi_1}(f(x_1)) = \mathcal{P}^f_{\pi_2}(f(x_2))$. Then after the third message flow, E can verify if any of x_1 and x_2 is the correct value to generate K. If x_1 (or x_2) is the correct value, then the corresponding password of π_1 (or π_2) must be the password shared between A and B. Otherwise, the two passwords, which generate π_1 and π_2, must not be the correct password. Hence no matter in which case, E can check at least two passwords in each impersonation. On the other hand, if E, impersonating B, constructs a pair (z, r_B) such that it obtains only one value x yielding $z = \mathcal{P}^f_\pi(f(x))$ for some $\pi \in \mathit{\Gamma}_{A,B,r_A,r_B}$, then this is not considered as a successful attack.

We consider this security requirement as a generalization of the 'associativity' problem discovered by Gong et al. in [7] and further exemplified by Jablon in [10]. The idea is also similar to the special characteristic of a password-entangled public-key generation primitive described in [8]. This limits the number of password guesses that the attacker can make to just one guess for each z, and for each run of a PAKE scheme.

Password-Keyed Permutation over $\mathbb{Z}_n^*$. In RSA-PAKE2, both $PwdK(k)$ and $\mathcal{S}(PK)$ are set to $\mathbb{Z}_n^*$. Hence the password-keyed permutation family of RSA-PAKE2 can be written as $\mathcal{P}^{(n,e)} : \mathbb{Z}_n^* \times \mathbb{Z}_n^* \to \mathbb{Z}_n^*$ and is defined as $(\pi, y) \to y \cdot \pi \bmod n$ for all $y, \pi \in \mathbb{Z}_n^*$. It is obvious that $\mathcal{P}^{(n,e)}$ is distinct. The following theorem says that it also satisfies Definition 1.

Theorem 1. *Given a RSA public key (n, e) such that the RSA Problem is hard, a password space PW, and a hash function $T : \{0,1\}^* \to \mathbb{Z}_n^*$ behaves like a random oracle, the password-keyed permutation $(\pi, y) \to y \cdot \pi \bmod n$ is secure for all $\pi, y \in \mathbb{Z}_n^*$.*

A proof is given in Appendix A.

4.3 The Generic PAKE Scheme

We now conclude the generalization and describe the entire generic scheme in the following. It is illustrated in Fig. 2.

The Generic PAKE Scheme

1. A generates a public key pair (PK, SK), picks $r_A \in_R \{0,1\}^k$ and sends PK and r_A to B.
2. B checks if PK is a valid public key. If it is invalid, B rejects the connection. Otherwise, it picks $r_B \in_R \{0,1\}^k$ and $s_B \in_R \mathcal{S}(PK)$, and computes $\pi = T(pw, A, B, r_A, r_B)$. It sends $z = \mathcal{P}_\pi(\mathcal{E}_{PK}(s_B))$ and r_B back to A, and destroys π from its memory.
3. A computes π accordingly and reveals the value of s_B from z. It generates a temporary symmetric key K and B's session key contribution c_B by $G_1(s_B)$ and $G_3(s_B)$, respectively. Then it picks its own session key contribution $c_A \in_R \{0,1\}^k$ and sends $(K \oplus c_A,\ G_2(K, c_A, A, B))$ to B. It later computes the session key σ as $G_4(c_A, c_B, A, B)$ and destroys s_B, π, c_A and c_B from its memory.
4. B computes K and c_B from s_B accordingly and destroys s_B from its memory. It reveals c_A from the first part of the incoming message and checks if the second part of the incoming message is $G_2(K, c_A, A, B)$. If it is false, B

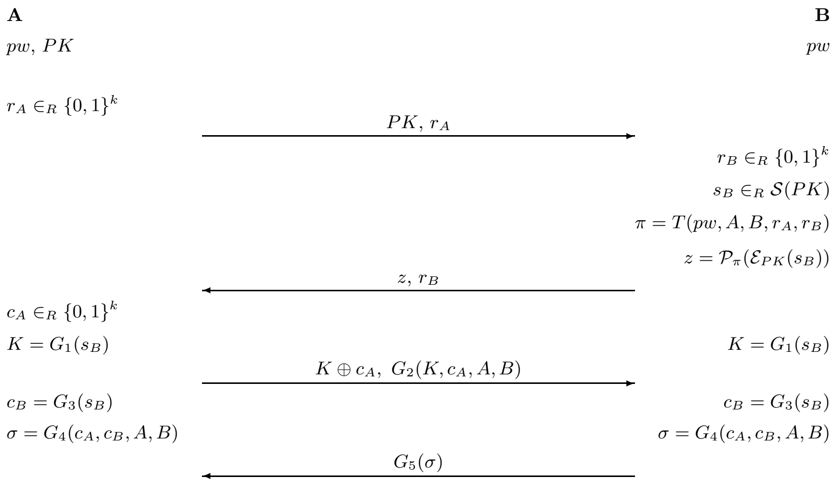
A
B
pw, PK
pw
$r_A \in_R \{0,1\}^k$
PK, r_A
$r_B \in_R \{0,1\}^k$
$s_B \in_R \mathcal{S}(PK)$
$\pi = T(pw, A, B, r_A, r_B)$
$z = \mathcal{P}_\pi(\mathcal{E}_{PK}(s_B))$
z, r_B
$c_A \in_R \{0,1\}^k$
$K = G_1(s_B)$
$K = G_1(s_B)$
$K \oplus c_A,\ G_2(K, c_A, A, B)$
$c_B = G_3(s_B)$
$c_B = G_3(s_B)$
$\sigma = G_4(c_A, c_B, A, B)$
$\sigma = G_4(c_A, c_B, A, B)$
$G_5(\sigma)$

Fig. 2. The Generic PAKE Scheme

rejects the connection. Otherwise, it computes σ accordingly and destroys c_A and c_B from its memory. $G_5(\sigma)$ is then sent back to A and the connection is accepted.

5. A checks if the incoming message is $G_5(\sigma)$. If it is true, A accepts the connection. Otherwise, it rejects the connection.

5 Concluding Remarks

The contributions of the paper can be divided into two parts. In the first part, we propose two modifications on a refined scheme of [17]. The modifications reduce the message size hence improve the network efficiency and memory footprint. More importantly, RSA-PAKE2 dramatically reduces the size of the message sent from the low-power client to the server. This saves a lot of battery power for a portable client from transmitting radio signal in wireless communications.

In the second part, we generalize RSA-PAKE1 and its modifications to a generic scheme and formalize the security requirements of all underlying primitives that the generic scheme is constituted. In the formalization, we introduce a new primitive called password-keyed permutation family to capture the features of the password related operations of RSA-PAKE1 and its modifications. We also show that these password related operations satisfy the security requirements of the password-keyed permutation family of the generic scheme if and only if the RSA problem is hard in the random oracle model. This also implies that RSA-PAKE2 is an instantiation of the generic scheme.

Other instantiations of the generic scheme are also possible. The most challenging problem of designing an instantiation is to devise a secure password-keyed permutation family $\mathcal{P}^{PK}$. For example, an interesting open problem is to devise one which is based on the Quadratic Residuosity Problem.

6 Acknowledgement

We thank Burt Kaliski for his valuable comments and suggestions in the preliminary work of this paper.

References

[1] Feng Bao. Security analaysis of a password authenticated key exchange protocol. to appear in Information Security (ISC 2003), Oct, 2003.

[2] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73, Fairfax, 1993. ACM.

[3] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password based protocols secure against dictionary attacks. In *Proceedings 1992 IEEE Symposium on Research in Security and Privacy*, pages 72–84. IEEE Computer Society, 1992.

[4] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In *Proc. EUROCRYPT 2000*, pages 156–171, 2000.

[5] Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes, and Cryptography*, 2(2):107–125, June 1992.

[6] R. Gennaro and Y. Lindell. A framework for password-based authenticated key exchange. In *Proc. EUROCRYPT 2003*. Springer-Verlag, 2003. Lecture Notes in Computer Science No. 2656, also in Cryptology ePring Archive: Report 2003/032.

[7] L. Gong, M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from guessing attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, 1993.

[8] IEEE. *P1363.2 / D10: Standard Specifications for Password-based Public Key Cryptographic Techniques*, Jul 2003.

[9] David P. Jablon. Strong password-only authenticated key exchange. *Computer Communication Review, ACM*, 26(5):5–26, 1996.

[10] David P. Jablon. Extended password key exchange protocols immune to dictionary attack. In *Proceedings of the WETICE'97 Workshop on Enterprise Security*, Cambridge, MA, USA, Jun 1997.

[11] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Proc. EUROCRYPT 2001*. Springer-Verlag, 2001. Lecture Notes in Computer Science No. 2045.

[12] Stefan Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Proc. of the Security Protocols Workshop*, pages 79–90, 1997. LNCS 1361.

[13] Philip MacKenzie, Sarvar Patel, and Ram Swaminathan. Password-authenticated key exchange based on RSA. In *Proc. ASIACRYPT 2000*, pages 599–613, 2000.

[14] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[15] Duncan S. Wong, Hector Ho Fuentes, and Agnes H. Chan. The performance measurement of cryptographic primitives on palm devices. In *Proc. of the 17th Annual Computer Security Applications Conference*, December 2001.

[16] Thomas Wu. The secure remote password protocol. In *1998 Internet Society Symposium on Network and Distributed System Security*, pages 97–111, 1998.

[17] Feng Zhu, Duncan S. Wong, Agnes H. Chan, and Robbie Ye. Password authenticated key exchange based on RSA for imbalanced wireless networks. In *Information Security (ISC 2002)*, pages 150–161. Springer-Verlag, September 2002. Lecture Notes in Computer Science No. 2433.

A Proof of Theorem 1

Proof. We show that breaking the password-keyed permutation $(\pi, y) \rightarrow y \cdot \pi \bmod n$ is computationally equivalent to solving the RSA Problem. By breaking the password-keyed permutation, we mean that given a RSA public key (n, e), a password space PW, and an ideal hash function $T : \{0,1\}^* \rightarrow \mathbb{Z}_n^*$, there exists a probabilistic polynomial-time algorithm (PPT) E^T such that for some $r_A \in \{0,1\}^k$, $A, B \in \{0,1\}^*$,

$$\begin{aligned}\Pr[\ & E^T(1^k, A, B, PW, (n,e), r_A) \rightarrow (z, r_B, x_1, x_2, \pi_1, \pi_2)\ :\ r_B \in \{0,1\}^k,\\ & z, x_1, x_2 \in \mathbb{Z}_n^*,\ \pi_1, \pi_2 \in \Gamma_{A,B,r_A,r_B},\ z \equiv x_1^e \cdot \pi_1 \equiv x_2^e \cdot \pi_2 \pmod n\ ]\\ & \geq \frac{1}{Q(k)}\end{aligned}$$

for some polynomial function Q.

Suppose that $\mathbf{Oracle}^{\mathbf{RSAP}}$ is a PPT for the RSA Problem with non-negligible probability. That is, given (n, e) and $y \in \mathbb{Z}_n^*$, it is non-negligible to compute $x \leftarrow \mathbf{Oracle}^{\mathbf{RSAP}}(n, e, y)$ such that $y \equiv x^e \pmod n$. We now construct a PPT E^T with access to a random oracle T to break the password-keyed permutation.

For a security parameter k, a server $A \in \{0,1\}^*$, a client $B \in \{0,1\}^*$, a password space PW, a RSA public key (n, e), and a k-bit binary string r_A, E^T proceeds as follows.

E^T = "On inputs 1^k, A, B, PW, (n, e), r_A,

1. randomly generates $r_B \in \{0,1\}^k$,
2. picks $pw_1, pw_2 \in PW$ and computes $\pi_1 = T(pw_1, A, B, r_A, r_B)$ and $\pi_2 = T(pw_2, A, B, r_A, r_B)$ by querying the random oracle of the form T.
3. Constructs two $x_1, x_2 \in \mathbb{Z}_n^*$ such that $x_1^e \cdot \pi_1 \equiv x_2^e \cdot \pi_2 \pmod n$. This is done by randomly pick an element $x_1 \in \mathbb{Z}_n^*$, define $z = x_1^e \cdot \pi_1 \bmod n$, and compute $x_2 \leftarrow \mathbf{Oracle}^{\mathbf{RSAP}}(n, e, z \cdot \pi_2^{-1} \bmod n)$.
4. Outputs z, r_B, x_1, x_2, π_1 and π_2."

It is easy to see that E breaks the password-keyed permutation with non-negligible probability.

Conversely, suppose that $\mathbf{Oracle}^{\mathcal{P}}$ is a PPT that breaks the password-keyed permutation $(\pi, y) \rightarrow y \cdot \pi \bmod n$ with non-negligible probability. We now show that a PPT C can be constructed to solve an instance of the RSA Problem with non-negligible probability, that is given a RSA public key (n, e) and an element $y \in \mathbb{Z}_n^*$, find $x \in \mathbb{Z}_n^*$ such that $y \equiv x^e \pmod{n}$.

Algorithm C uses $\textbf{Oracle}^{\mathcal{P}}$ as a black box, but has full control over its oracles. That is, C simulates $\textbf{Oracle}^{\mathcal{P}}$'s view by providing 1^k, $A, B \in \{0,1\}^*$, a password space PW, (n, e) as A's public key, $r_A \in_R \{0,1\}^k$ and answers for queries of the form T. In a non-negligible case when $\textbf{Oracle}^{\mathcal{P}}$ successfully generates $(z, r_B, x_1, x_2, \pi_1, \pi_2)$ such that $z \equiv x_1^e \cdot \pi_1 \equiv x_2^e \cdot \pi_2 \pmod{n}$ for $\pi_1, \pi_2 \in \mathit{\Gamma}_{A,B,r_A,r_B}$, $\textbf{Oracle}^{\mathcal{P}}$ queries with the form T for at least twice to generate π_1 and π_2 with probability at least $1 - 1/\phi(n)^2$. Suppose C guesses correctly on these two queries. Then C picks $r \in_R \mathbb{Z}_n^*$, and provides $r^e \bmod n$ and y as answers. For all other queries of the form T, C simply picks random elements in $\mathbb{Z}_n^*$ as answers.

Without loss of generality, we assume that $\pi_1 = r^e \bmod n$ and $\pi_2 = y$. After $\textbf{Oracle}^{\mathcal{P}}$ generates $(z, r_B, x_1, x_2, \pi_1, \pi_2)$, we have

$$z \equiv x_1^e \cdot r^e \equiv x_2^e \cdot y \pmod{n}$$

$$y \equiv (x_1 \cdot x_2^{-1} \cdot r)^e \pmod{n}$$

Hence

$$x \equiv x_1 \cdot x_2^{-1} \cdot r \pmod{n}.$$

For C to make correct guesses of the two queries of the form T that generate π_1 and π_2, we can see that if C makes two random guesses, the probability of guessing correctly is at least $1/Q^2$ where Q is the total number of queries made by $\textbf{Oracle}^{\mathcal{P}}$. Therefore, C solves the instance of RSA Problem with probability at least $(1 - 1/\phi(n)^2)/Q^2$ of the success probability of $\textbf{Oracle}^{\mathcal{P}}$. □