Original version will appear in the Proc. of ISC 2002 (Sep. 30 - Oct. 2, 2002, Sao Paulo, Brazil). In LNCS2433 series.

Password Authenticated Key Exchange based on RSA for Imbalanced Wireless Networks

Feng Zhu, Duncan S. Wong, Agnes H. Chan and Robbie Ye

College of Computer Science
Northeastern University
Boston, MA 02115
{zhufeng, swong, ahchan, robbieye}@ccs.neu.edu

Abstract. We consider an imbalanced wireless network setup in which a low-power client communicates with a powerful server. We assume that public key cryptographic operations such as Diffie-Hellman key exchange conducted over a large multiplicative group is too computationally intensive for a low-power client to implement. In this paper, we propose an authenticated key exchange protocol such that it is efficient enough to be implemented on most of the target low-power devices such as devices in sensor networks, smart cards and low-power Personal Digital Assistants. In addition, it is secure against dictionary attacks. Our scheme requires 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 8 kbps. The computation time can be improved to 300 msec and the transmision time can also be reduced to 300 msec if caching is allowed.

1 Introduction

Classical symmetric key based authenticated key exchange protocols dictate the shared cryptographic keys to be long enough and randomly generated so that they can deter key-guessing attacks, or so-called brute force key searching attacks. They are widely used to provide mutual entity authentication in conjunction with session key establishment. However, it is very difficult to memorize such keys and therefore these protocols usually have little implication of providing user authentication. Furthermore, those schemes require the keys and some other participant-specific information to be stored in some tamper-proof security module at each entity. For applications where a security module is not available or user authentication is prevailing, authenticated key exchange protocols which are secure against key-guessing attacks are needed. Protocols designed to provide mutual authentication and key exchange, while also secure against offline key-guessing attacks or the so-called offline dictionary attacks, are called Password Authenticated Key Exchange protocols.

The objective of a password authenticated key exchange protocol can be described as follows. After two communicating parties execute the scheme and when both parties terminate and accept, 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 its intended partner and the key is derived from contributions of both parties (key exchange). These goals are essentially the same as those of cryptographic key based authenticated key exchange schemes. But unlike the cryptographic key based authenticated key exchange schemes, these two communicating parties do not have any pre-shared cryptographic symmetric key, certificate or information of any trusted third party. Instead they only have a password shared between them. Usually the password is picked from a relatively small password space which can be enumerated efficiently by an adversary.

We focus on designing a password authenticated key exchange scheme which is suitable for implementation on a type of distributed network, called imbalanced (asymmetric) network. An imbalanced network consists of two sets of entities, namely a set of powerful servers and a set of low-power clients. The servers are having similar computational power and memory capacity to current desktop machines while the clients are only comparable to those microprocessor-based smart cards, wearable devices in a PAN (Personal Area Network), low-end PDAs (Personal Digital Assistants) or cellular phones. The scheme is going to be designed for communications between a client and a server. In addition, we assume that the bandwidth of the network is low. A typical cellular network in which a mobile unit communicating with a base station, and a Bluetooth-based PAN in which an earpiece communicating with a handset are two typical examples of imbalanced networks.

1.1 Related Results

Since the first set of password authenticated key exchange schemes called EKE was proposed by Bellovin and Merritt [2] in 1992, there have been many new proposals suggested in recent years [6, 10, 15, 3, 11, 7]. A more comprehensive list of these schemes can be found in [8] and Jablon's research links[1]. Most of these schemes are based on Diffie-Hellman key exchange. In an imbalanced network however, the large modular exponentiation operation, which needs to be carried out by both communicating parties, may take a long time for a low-power device to compute. As an example, one modular exponentiation taken in a 512-bit cyclic subgroup with 160-bit prime order spends 27 seconds to complete when running on a 16MHz Palm V [13]. This becomes very noticeable when the client is a smart card, a low-end PDA or a tiny wearable Bluetooth device embedded with a very limited microcontroller.

In [2], the authors investigated the feasibility of using RSA to instantiate the EKE protocol. If the RSA public exponent is short, the encryption operation can be done efficiently and the corresponding protocol may be very suitable for the imbalanced network. However, they pointed out that an e-residue attack may be feasible if the receiving party has no way to verify whether the 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 n where n is the RSA modulus. To thwart this attack, the authors considered an interactive protocol to allow a

[1] http://www.integritysciences.com/links.html

receiver to detect a fraudulent value of the public exponent. However, we find that the interactive protocol is insecure. In Sect. 2, we describe an attack against the interactive protocol. We also propose a modified version such that it prevents the attack and at the same time achieves better efficiency when compared with the original one.

In 1997, Lucks proposed a scheme called OKE (Open Key Exchange) [10] which is based on RSA. It was later found to be insecure against a variant of e-residue attacks due to MacKenzie, et al. [11]. In [11], the authors modified OKE such that a *large* prime is chosen to be the RSA public exponent. This ensures that the public exponent is relatively prime to $\phi(n)$. On the other hand, this defeats the purpose of using RSA for low-power clients in an imbalanced network because the computational complexity of the protocol is no less than most of the protocols based on Diffie-Hellman key exchange.

Another issue of these schemes is that all of them are actually providing key *transport* instead of key exchange, namely the secrecy of the final session key depends on the secrecy of a session key contribution generated by one party only instead of both parties. Hence if that party's session key contribution is compromised, the session key is compromised. In the key exchange case however, the final session key is derived from the session key contributions of both parties. Compromising only one party's session key contribution does not compromise the final session key. Therefore in our work, we prefer to design a password authenticated key *exchange* scheme.

1.2 Overview of Our Results

We propose a password authenticated key exchange protocol based on RSA scheme with short public exponents. We show how e-residue attacks can be prevented by using a modified interactive protocol. We also evaluate its performance when implemented on a low-power client such as a 16MHz Palm V and discuss its efficiency on data transmission.

The subsequent sections are organized as follows. In the next section, we describe an e-residue attack against an instantiation of EKE protocol called RSA-EKE. We modify the corresponding interactive protocol and show that the modified version can both prevent the e-residue attack and improve the network efficiency. In Sect. 3, we propose our RSA-based password authenticated key exchange scheme and specify the security requirements of each component in the scheme. We then evaluate the performance of the scheme in Sect. 4 and discuss the feasibility of implementing the scheme on a low-end PalmPilot such as a 16MHz Palm V. Finally in Sect. 5, we conclude the paper with some further remarks.

2 The e-residue Attack and the Interactive Protocol

In this section we describe an instantiation of EKE protocol based on RSA, abbreviated as RSA-EKE, and explain its vulnerability against an e-residue attack.

For simplicity of description, we adopt the following notations throughout the paper.

Let

- A be a server and B be a low-power client.
- ID_A and ID_B denote the identities of A and B respectively.
- $r \in_R \{0,1\}^l$ denote a random binary string r of length l, and $\{0,1\}^l$ denote the set of all l-bit binary strings.
- pw denote a password shared between A and B. It is picked from a password space PW according to certain probability distribution.
- E_K, D_K be a symmetric encryption algorithm and respectively the decryption algorithm defined by a symmetric key $K \in \{0,1\}^l$.
- H, T, G_1, G_2, G_3 and h be distinct cryptographic hash functions.
- (n, e) be a RSA public key pair such that n be the public modulus and e be the public exponent.

The RSA-EKE protocol without the interactive protocol is illustrated in Fig. 1. In the protocol, A and B are sharing a password pw. R is a random

	A		**B**
	pw		pw
1.		$\xrightarrow{ID_A,\ n,\ e}$	
2.		$\xleftarrow{E_{pw}((R, challenge_B)^e \bmod n)}$	
3.		$\xrightarrow{E_R(challenge_B, chanllenge_A)}$	
4.		$\xleftarrow{E_R(chanllenge_A)}$	

Fig. 1. RSA-EKE without Interactive Protocol

session key, and $challenge_A$ and $challenge_B$ are two random numbers used to authenticate B and A respectively.

In [2], several issues on using RSA have been pointed out. One of the related issues raised by the authors is the e-residue attack.

2.1 e-residue Attack

To ensure that the RSA scheme works correctly, the public exponent e has to be relatively prime to $\phi(n)$. An active attacker can launch an e-residue attack by impersonating the server A and sending (n', e') to the client B where $(e', \phi(n')) \neq 1$. The encryption function $x \to x^{e'} \bmod n'$ is no longer a permutation on $\mathbf{Z}_{n'}$. The encryption function maps elements to the set of e'-residues, which is only a proper subset of $\mathbf{Z}_{n'}$. Since the adversary knows the factorization of n', it is easy for the adversary to check whether an element $a \in \mathbf{Z}_{n'}$ is an e'-residue. Hence after receiving $E_{pw}(y)$ in step 2, where $y = (R, \mathit{challenge}_B)^{e'} \bmod n$, the adversary can use a trial password pw' to obtain $y' = D_{pw'}(E_{pw}(y))$ and checks whether y' is an e'-residue or not. If it is not, then the trial password pw' is not the correct password and therefore can be removed from the password space PW.

Obviously this attack would not work if B can determine whether e is relatively prime to $\phi(n)$. So the following problem is crucial to resist the e-residue attack.

Problem: Given two composite n and e such that $3 \leq e < n$. Determine if e is relatively prime to $\phi(n)$.

In [12], the authors pointed out that it is not known whether there exists a polynomial-time algorithm to solve this problem in general. In [11], the authors proposed two methods to enforce the relative primality of $\phi(n)$ and e. The first method is to set e to be a prime greater than n so to guarantee that $(e, \phi(n)) = 1$. The second method is to set e to be a prime such that it is greater than $\sqrt{n}$ and $(n \bmod e)$ does not divide n. It can be shown that this also guarantees that e is relatively prime to $\phi(n)$ [9]. The drawback for these two methods is that the value of e is relatively large; thereby defeats the advantage of using a small e to improve the performance. To attain high efficiency of computation for the low power clients, small public exponents are needed. In order to thwart e-residue attacks, interactive protocols based on challenge and response may be used.

2.2 An Interactive Protocol

We now revisit a preliminary idea mentioned in [2] in detecting fraudulent values of e. The idea is based on the fact that for odd integer n and e ($e \geq 3$) such that $(e, \phi(n)) \neq 1$, any e-residue modulo n should have at least 3 e-th roots. As described in [2], a client B can verify e interactively, by asking A to decrypt a number of encrypted messages under (n, e). Specifically B picks N integers $m_i \in_R \mathbf{Z}_n$, where N is a security parameter (say 10) and sends $\{c_i \equiv m_i^e \pmod n\}_{1 \leq i \leq N}$ to A. A computes the e-th root of each c_i as m_i' and sends $\{m_i'\}_{1 \leq i \leq N}$ back.

For $1 \leq i \leq N$, if $(e, \phi(n)) = 1$, each c_i should have a unique e-th root and $m_i' = m_i$. On the other hand, if $(e, \phi(n)) \neq 1$, each c_i should have at least 3 e-th roots and they are equally likely. The probability that A guesses all the N e-th roots correctly is therefore at most 3^{-N}.

Although the authors of [2] could not find any immediate weakness in the RSA-EKE, they were correct in being concerned about the security of the protocol being compromised by an intruder using a server A as a decryption oracle. In the following, we show how an attacker E can impersonate B and use this decryption oracle in guessing the password shared between A and B.

The RSA-EKE with the interactive protocol is illustrated in Fig. 2. Suppose

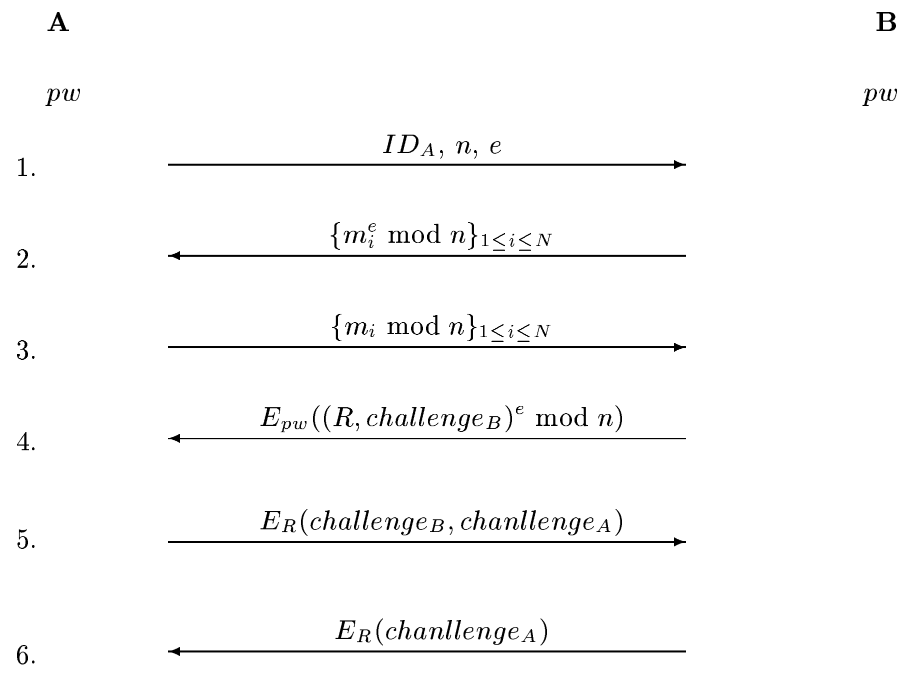

Fig. 2. RSA-EKE with Interactive Protocol

that during one successful run of the protocol between A and B, an adversary E gets the messages of the fourth and fifth flows, that is $msg_4 = E_{pw}((R,$ *challenge*$_B)^e \bmod n)$ and $msg_5 = E_R(challenge_B,\ challenge_A)$ respectively. Assume that the same public key (n, e) is used for multiple sessions. Then E can impersonate B and communicate with A by randomly chooses N trial passwords $\{pw'_i\}_{1 \le i \le N}$ from the password space PW. It then impersonates B and communicates with A. In the second flow, it sends $\{D_{pw'_i}(msg_4)\}_{1 \le i \le N}$ to A while expecting $\{(D_{pw'_i}(msg_4))^d \bmod n\}_{1 \le i \le N}$ in reply. We can see that for any i $(1 \le i \le N)$, if $pw'_i = pw$, then the corresponding reply must contain the correct value of R and $challenge_B$. E can check this via msg_5. Hence if the checking fails, pw must be a wrong guess and E then removes it from PW. Thus E can check N passwords in one impersonation.

This attack can be prevented if the server A cannot be used as a decryption oracle. This can be accomplished by modifying it slightly via adding a cryptographic hash function. In the next section, we propose the modification.

3 Our Protocol

We now describe a new password authenticated key exchange protocol based on RSA for imbalanced networks. Our objectives include making computational complexity at the client side as low as possible, requiring minimum memory requirement and reducing the size of data transmission. The protocol is shown in Fig. 3 and described as follows.

In the protocol,

- A and B are sharing a password $pw \in PW$.
- A generates a RSA public key pair $(n,\, e)$ using a public key generator and selects $r_A \in_R \{0,1\}^l$. She sends $(n,\, e)$ and r_A to B.
- B checks if $(n,\, e)$ is a valid public key by using an Interactive Protocol. If it is invalid, B rejects the connection. Otherwise, it first picks $r_B \in_R \{0,1\}^l$, $s_B \in_R \mathbf{Z}_n$. Then it computes $\pi = T(pw, \mathrm{ID}_A, \mathrm{ID}_B, r_A, r_B)$ where ID_A and ID_B are the identification information of A and B, respectively and $T : \{0,1\}^* \rightarrow \mathbf{Z}_n$ is a cryptographic hash function. It sends r_B and $z = s_B^e + \pi \bmod n$ to A. B destroys π from its memory.
- A computes the corresponding π and obtains the value of s_B from the second message of the protocol. It then generates a temporary symmetric encryption key K and B's session key contribution c_B by computing $G_1(s_B)$, and $G_2(s_B)$, respectively where $G_1 : \{0,1\}^* \rightarrow \{0,1\}^l$ and $G_2 : \{0,1\}^* \rightarrow \{0,1\}^l$ are two distinct cryptographic hash functions. It then picks its own session key contribution $c_A \in_R \{0,1\}^*$ and sends $E_k(c_A, \mathrm{ID}_B)$ to B. It later computes the session key σ as $G_3(c_A, c_B, \mathrm{ID}_A, \mathrm{ID}_B)$ and destroys s_B, c_A, c_B and K from its memory.
- B computes K and c_B from s_B accordingly. It decrypts the incoming message and checks if it contains its own identity ID_B along with some l-bit binary string. If it is false, B terminates the protocol run with failure. Otherwise, it denotes the l-bit binary string as c'_A and uses it as A's session key contribution and computes the session key σ' accordingly. It then destroys s_B, c'_A ,c_B from its memory. $h(\sigma')$ is then sent back to A and the connection is accepted.
- A checks if the incoming message $h(\sigma') = h(\sigma)$. If it is true, then A accepts the connection. Otherwise, it terminates the protocol run with failure.

To ensure our protocol is applicable to the imbalanced networks in terms of efficiency, we need to use short RSA public exponents. To resist e-residue attacks,

	A		**B**
	pw		pw
1.	$r_A \in_R \{0,1\}^l$	$\xrightarrow{n,\ e,\ r_A}$	
2.		$\xleftarrow{\{m_i^e \bmod n\}_{1 \le i \le N}}$	$\{m_i \in_R \mathbf{Z}_n\}_{1 \le i \le N}$
3.		$\xrightarrow{\{H(m_i')\}_{1 \le i \le N}}$	
4.			$H(m_i') \stackrel{?}{\equiv} H(m_i), 1 \le i \le N$
			$r_B \in_R \{0,1\}^l$
			$s_B \in_R \mathbf{Z}_n$
			$\pi = T(pw, \mathrm{ID}_A, \mathrm{ID}_B, r_A, r_B)$
			$z = s_B^e + \pi \bmod n$
		$\xleftarrow{z,\ r_B}$	
5.	$c_A \in_R \{0,1\}^l$		$K = G_1(s_B)$
	$K = G_1(s_B)$		$c_B = G_2(s_B)$
		$\xrightarrow{E_K(c_A,\ \mathrm{ID}_B)}$	
6.	$c_B = G_2(s_B)$		$\sigma' = G_3(c_A, c_B, \mathrm{ID}_A, \mathrm{ID}_B)$
	$\sigma = G_3(c_A, c_B, \mathrm{ID}_A, \mathrm{ID}_B)$	$\xleftarrow{h(\sigma')}$	
7.	$\sigma' \stackrel{?}{\equiv} \sigma$		

Fig. 3. Password Authenticated Key Exchange Using RSA

our scheme includes an interactive protocol in step 2 and 3 shown in Fig. 3 by having A returning hash values of the decrypted challenge numbers, that is $\{H(m_i)\}_{1 \le i \le N}$. The hash function H is defined as $H : \{0,1\}^* \to \{0,1\}^{Q'(l)}$ where Q' is some polynomial function. We conjecture that in this way, A can no longer be used as a decryption oracle in helping an adversary to invert RSA.

In Fig. 3, z is computed as an encrypted random element in $\mathbf{Z}_n$ followed by a modular addition of π where π is a function of the password pw with some nonces r_A and r_B and identification information. In fact, the RSA encryption algorithm can be treated to be a trapdoor one-way permutation over $\mathbf{Z}_n$ and we can also consider the modular addition operation as a permutation over the same set of elements.

Once we generalize these two operations as trapdoor permutation and permutation over the same set of elements, we may be able to apply them to other methods with the associativity issue [5] considered. Here below are the two other methods.

Method 1. By specifying the RSA encryption algorithm performed over the multiplicative group $\mathbf{Z}_n^*$, that is by choosing $s_B \in_R \mathbf{Z}_n^*$, we can compute z as $s_B^e \cdot \pi \bmod n$. Hence the 'password-keyed' permutation now becomes a simple modular multiplication and the protocol is similar to [10, 11].

Method 2. Let $T : \{0,1\} \to \{0,1\}^{|n|-1}$ and the password-keyed permutation be defined as $z = y \oplus \pi$ for all $y \in \{0,1\}^{|n|-1}$. Since more than half of the elements in $\mathbf{Z}_n$ is smaller than $2^{|n|-1}$, the expected number of times the RSA encryption needs to perform before picking a $s_B \in_R \mathbf{Z}_n$ with $|s_B^e \bmod n| \leq |n|-1$ is less than 2. On the efficiency of this method, we can see that with greater than half probability, the client needs to carry out the RSA encryption once (instead of twice or more). Since the password-keyed permutation requires only one ($|n|$-1)-bit XOR operation, this method is more efficient than the previous two in the majority case. However, with less than half probability, this scheme is expected to carry out the RSA encryption twice, which takes one more modular multiplication if e = 3 when compared with the previous methods.

We now review the first method we proposed, namely using modular addition as a password-keyed permutation and the set of elements that the RSA encryption algorithm is defined over is $\mathbf{Z}_n$. This method when compared with the other two allows the client to carry out only one RSA encryption and it is deterministic. Furthermore since the password-keyed permutation requires only one $|n|$-bit addition operation, this method is more efficient than the other two in the majority cases.

A secure symmetric encryption scheme in our protocol can be instantiated by most of the well-known block ciphers or stream ciphers. It is also straightforward to construct one from a given pseudorandom function family. Let $f : Keys(l) \times Dom(l) \to \{0,1\}^{Q''}(l)$ be a pseudo-random function family where $Q''(l)$ is some polynomial function. The symmetric encryption scheme can be defined as $E_K(x) = (r, f(K,r) \oplus x)$ for all $x \in \{0,1\}^{Q''(l)}$ where $r \in_R Dom(l)$. Alternatively, we can use some appropriate hash function that behaves like a random oracle [1], $H_4 : \{0,1\}^{2l} \to \{0,1\}^{Q''}(l)$ and define the symmetric encryption scheme as $E_K(x) = (r, H_4(K,r) \oplus x)$. This construction allows a target application to build up the encryption scheme from a common hash function which may have already been used to construct T, G_1, G_2, G_3, H and h instead of implementing a block cipher or a stream cipher.

4 Efficiency

In an imbalanced network, the servers are generally assumed to be powerful, and all the operations of the key exchange scheme can be carried out at the server side efficiently. Hence we focus on evaluating the efficiency at the client side in terms of computational complexity as well as network and storage requirements. First we look into the efficiency issue and show that the interactive protocol is efficient enough in our target applications if some appropriate value of N is chosen.

4.1 The Interactive Protocol

One concern of this interactive protocol is that it is expensive in terms of the amount of messages sent and the number of encryptions needed to be carried out by the low-power client.

On the amount of messages needed to be sent, this interactive protocol adds two more flows and the size of the first flow of the interactive protocol is $N|n|$ bits. Suppose $N = 10$ and $|n| = 512$ bits, this flow takes about 640 msec to transmit if the network throughput is 8kbps. The second flow added by the interactive protocol is a sequence of hash values. If H maps an input to a binary string of 160-bit long, 200 bytes are needed to be sent. Therefore this interactive protocol adds about 840 msec to the total data transmission time of the scheme.

4.2 Computational Complexity

With respect to the computation complexity, B needs to compute N modular exponentiations with the exponent e. If e is small and has a small number of ones in its binary representation, the total number of modular multiplications/squares may still be relatively small when compared with schemes based on Diffie-Hellman key exchange. For example, if $e = 3$ and $|n| = 512$, one encryption essentially takes two 512-bit modular multiplications. As each 512-bit modular multiplication takes about 107 msec when implemented on a 16MHz Palm V [14], the encryptions for preparing the first flow of the interactive protocol take about 2.14 seconds.

With respect to the decryption, it is widely known that each operation can be done within 5 milliseconds on a machine with 1GHz Intel processor. Hence the interactive protocol can barely make any significant impact on the performance at the server side.

As a summary on the computational complexity at the client side, it needs to perform $2(N + 1)$ modular multiplications, one modular addition and $N + 5$ hashes. As we measured on a 16MHz Palm V, one 512-bit modular addition takes 3.28 msec to complete and less than 4.6 msec is required to digest 80 bytes of data if hash functions are based on SHA-1. Therefore, the scheme including the interactive protocol takes less than 2.5 seconds of pure computation.

4.3 Network and Storage Efficiency

There are six message flows in a single run of the protocol. As mentioned above, the interactive protocol spends 840 msec to transmit on a 8kbps network. By including the time for transmitting other flows, the scheme takes a total of about one second for transmission.

On the storage requirement, the client needs to have $N \cdot |n|$ bits of dynamic memory for storing $\{m_i^e \bmod n\}_{1\le i\le N}$ and another $NQ'(l)$ bits of memory for storing $\{H(m_i)\}_{1\le i\le N}$ if the second flow is sent to A in one single message flow. This implies that the low-power device should have about 840 bytes of dynamic memory for data. For those devices which do not have that much of data memory, the second flow can be subdivided into several flows so that only a portion of $\{m_i^e \bmod n\}_{1\le i\le N}$ and $\{H(m_i)\}_{1\le i\le N}$ are kept in the data memory at a time.

4.4 Further Optimization — Public Key Caching

In some situation, B may cache $(n,\, e)$ for multiple sessions after one run of the interactive protocol. Hence it can save about 2.98 seconds for session key establishment in each of the subsequent sessions. Then each subsequent session only needs 300 msec of pure computation and its network efficiency is also improved to less than 300 msec.

In general, A may publish its RSA public key in advance or use the same public key to communicate with other participants besides B, without risking its security. In addition, this allows the clients to cache the value of (n, e) in their local memory and saves them from conducting the interactive protocol with A in any further runs of the protocol.

An even prominent gain of caching A's public key is to allow the client to further improve performance via pre-computation. As shown in the previous section, if (n, e) is cached, B can randomly pick s_B and precompute $y = x^e \bmod n$ before running the protocol. This allows B to reduce 214 msec of runtime on a PalmPilot. If r_B and the corresponding K and c_B are also pre-generated, this leaves B to carry out only three hashes, one $|n|$-bit modular addition and one symmetric key decryption during the runtime of the protocol. These operations require a total of less than 30 milliseconds of computation time when implemented on a 16MHz Palm V with some suitable symmetric key decryption algorithm such as Rijndael [4]. The obvious limitation of deploying this optimization is the memory requirement of storing A's public key and those pre-computed values. We notice that this requires $(2|n| + 4l + |e|)/8 \approx 212$ bytes of data to be stored, which may not fit in some extremely memory-limited devices such as in sensor networks. On the other hand, it is very suitable for applications running on smart card systems, wearable devices and PalmPilots.

5 Conclusion Remarks

It is clear that if the server's private key is compromised, an adversary can launch offline dictionary attack to find out the password and get all the s_B's of previous sessions. However, knowing just the password may not help reveal the session keys of previous sessions. Hence our scheme provides half forward secrecy. Since the client may be a weak device while the server can support much stronger security measures than the client, we believe that forward secrecy on the client side is a crucial feature.

Under the security requirements for each component of the scheme which are specified in the previous sections, we can show, in [13], that the scheme is secure under the random oracle model [1].

References

1. 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.
2. 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.
3. Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *Proc. EUROCRYPT 2000*, pages 156–171, 2000.
4. J. Daemen and V. Rijmen. AES proposal: Rijndael. *AES Algorithm Submission*, Sep 1999. http://www.nist.gov/aes.
5. 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.
6. David P. Jablon. Strong password-only authenticated key exchange. *Computer Communication Review, ACM*, 26(5):5–26, 1996.
7. 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.
8. Taekyoung Kwon. Ultimate solution to authentication via memorable password. *Contribution to the IEEE P1363 Study Group*, May 2000.
9. H. W. Lenstra, Jr. Divisors in residue classes. *Mathematics of Computation*, 42(165):331–340, 1984.
10. 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.
11. Philip MacKenzie, Sarvar Patel, and Ram Swaminathan. Password-authenticated key exchange based on RSA. In *Proc. ASIACRYPT 2000*, pages 599–613, 2000.
12. Philip MacKenzie and Ram Swaminathan. Secure network authentication with password identification. Submitted to IEEE P1363a, 1999.
13. Duncan S. Wong. On the design and analysis of authenticated key exchange schemes for low power wireless computing platforms. Ph.D. Thesis, July 2002.
14. 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*, Dec 2001.
15. Thomas Wu. The secure remote password protocol. In *1998 Internet Society Symposium on Network and Distributed System Security*, pages 97–111, 1998.