On the Design of RSA with Short Secret Exponent*

Hung-Min Sun[1], Wu-Chuan Yang[2] and Chi-Sung Laih[2]

[1] Department of Computer Science and Information Engineering
National Cheng Kung University; Tainan, Taiwan 701
hmsun@mail.ncku.edu.tw
[2] Department of Electrical Engineering
National Cheng Kung University, Tainan, Taiwan 701
wcyang77@ms32.hinet.net
laihcs@eembox.ee.ncku.edu.tw

Abstract. At Eurocrypt'99, Boneh and Durfee presented a new short secret exponent attack which improves Wiener's bound ($d< N^{0.25}$) up to $d< N^{0.292}$. In this paper we show that it is possible to use a short secret exponent which is below these bounds while not compromising with the security of RSA provided that p and q are differing in size and are large enough to combat factoring algorithms. As an example, the RSA system with d of 192 bits, p of 256 bits, and q of 768 bits is secure against all the existing short secret exponent attacks. Besides, in order to balance and minimize the overall computations between encryption and decryption, we propose a variant of RSA such that both e and d are of the same size, e.g., $\log_2 e \approx \log_2 d \approx 568$ for a 1024-bit RSA modulus. Moreover, a generalization of this variant is presented to design the RSA system with $\log_2 e + \log_2 d \approx \log_2 N + l_k$ where l_k is a predetermined constant, e.g., 112. As an example, we can construct a secure RSA system with p of 256 bits, q of 768 bits, d of 256 bits, and e of 880 bits.

1 Introduction

The RSA public-key cryptosystem was invented by Rivest, Shamir, and Adleman [16] in 1978. Since then, the RSA system has been the most well-known and accepted public key cryptosystem. Usually, the RSA system is deployed in various application systems for providing privacy and/or ensuring authenticity of digital data. Hence many practical issues have been sequentially considered when implementing RSA, e.g., how to reduce the storage for RSA modulus [13,24], how to use short public exponent for reducing the encryption execution time (or signature-verification time) [2-4,8-9], and how to use short secret exponent for reducing the decryption execution time (or signature-generation time) [1,25-26]. In this paper, we are interested in the use of short secret exponent because it is particularly advantageous when there is a large difference in computing power between two communicating devices. For

* This wok was supported in part by the National Science Council, Taiwan, under contract NSC-88-2213-E-324-007 and NSC88-2213-E-006-025.

K. Y. Lam, E. Okamoto and C. Xing (Eds.): ASIACRYPT'99, LNCS 1716, pp. 150-164, 2000.

example, it would be desirable for a smart card to have a short secret exponent in order to speedup the decryption or the generation of signatures in the smart card, and for a larger computer to have a short public exponent in order to speedup the encryption or the verification of signatures required in the smart card. We are also interested in the use of balanced and minimized public and secret exponents that the length of both is approximately equal and is as short as possible. The main motivation for this is to provide the requirement of those applications when the computing power between two communicating devices is approximately equal. Particularly, it is advantageous when a sequence of encryptions and decryptions (or signature generations and verifications) are required to run synchronously, i.e., no party is idle between communication. Inspired by the above concept, we are also interested in balancing and minimizing the encryption time and the decryption time when there is a difference in computing power between two communicating devices. It is an intuitive thought that if the computation amount in encryption is heavier, then the computation amount in decryption will be lighter, and vice versa. Therefore there should exists a trade-off between encryption and decryption, e.g., the overall computation amount is constant. Consequently, what we concern is how to reduce the overall computations used in encryption and decryption and how to distribute the overall computations between encryption and decryption. If the distributed computations between encryption and decryption are roughly proportional to the computing power of the two communicating devices, we can balance the encryption time and the decryption time even if there is a difference in computing power between these two devices.

We first describe a simplified version of RSA primitive as follows: Let $N=pq$ be the product of two large primes. If both p and q are 512 bits long, then N is about 1024 bits long. Let e and d be two integers satisfying $ed=1 \bmod \phi(N)$, where $\phi(N)=(p-1)(q-1)$ is the Euler totient function of N. Here we call N the *RSA modulus*, e the *public exponent*, and d the *secret exponent*. The public key is the pair (N, e) and the secret key is d. For simplicity, we assume the owner of the secret key is Alice. To provide privacy, one can encrypt a message M into a ciphertext C by: $C= M^e \bmod N$, while only Alice can decrypt the ciphertext C into the plaintext M by: $M= C^d \bmod N$. To ensure authenticity of digital data, only Alice can sign a document M to obtain a signature S by: $S= M^d \bmod N$, while one can verify the validity of Alice's signature S on M by checking if $M= C^e \bmod N$ satisfies a predetermined redundancy scheme.

For a fixed modulus size, the RSA encryption or decryption time is roughly proportional to the number of bits in the exponent. To reduce the encryption time (or the signature-verification time), one may wish to use a small public exponent e. The smallest possible value for e is 3. If $e=3$ is used for encryption, it has been proven to be insecure against some short public exponent attacks [9]. The most powerful attack on short public exponent is due to Coppersmith, Franklin, Patarin and Reiter [4]. Under their attack, the RSA primitive is insecure for all public exponents of length up to around 32 bits. Therefore it is suggested to use public exponents of length more than 32 bits. Note that these short public exponent attacks succeed only in the encryption of the RSA primitive. They cannot work in the RSA with the protection of the standards PKCS#1 v2.0 or IEEE P1363.

On the other hand, to reduce the decryption time (or the signature-generation time), one may also wish to use a small secret exponent d. Unfortunately, based on the

convergents of the continued fraction expansion of a given number, Wiener [26] showed that the RSA system can be totally broken if $d< N^{0.25}$. Verheul and van Tilborg [25] proposed an extension of Wiener's attack that allows the RSA system to be broken when d is a few bits longer than $0.25\log_2 N$. For $d> N^{0.25}$, their attack need do an exhaustive search for about $2t$+8 bits, where $t=\log_2(d / N^{0.25})$. If t =20 (which leads to an order of magnitude 2^{48}) is feasible to do an exhaustive search, then the RSA system with $d< 2^{20} N^{0.25}$ is insecure. Thus this gives a 20 bits improvement on Wiener's bound. Recently, based on lattice basis reduction, Boneh and Durfee [1] proposed a new attack on the use of short secret exponent. They improved Wiener's bound up to $d< N^{0.292}$. This gives a 43 bits improvement on Wiener's bound if N is the size of 1024 bits. In general, the use of short secret exponent encounters a more serious security problem than the use of short public exponent.

In this paper, we show that it is possible to use a short secret exponent which is below both Wiener's bound and Boneh and Durfee's bound while not compromising the security of RSA provided that p and q are differing in size and are large enough to combat the factoring algorithms which are based on elliptic curves. As an example, when p is the size of 256 bits and q is the size of 768 bits, d of 192 bits is large enough to combat the existing short secret exponent attacks. In this study of the balanced and minimized public and secret exponents, we propose a secure variant of RSA such that e and d are of the same size, e.g., $\log_2 e \approx \log_2 d \approx 568$ for a 1024-bit RSA modulus. We analyze the security of the proposed RSA variant according to the ways of attacking short secret exponent RSA and conclude that the proposed scheme is secure enough to defeat all the existing short secret exponent attacks. Finally, the trade-off between the length of secret exponent and public exponent is analyzed. We show that it is possible to design a secure RSA system with $\log_2 e+\log_2 d \approx \log_2 N+l_k$ where l_k is a predetermined constant, e.g., 112. Compared with typical RSA system that e is of the same order of magnitude as N if d is first selected, these variants of RSA have the advantage that the overall computations can be significantly reduced. As an example, we can construct a secure RSA system with p of 256 bits, q of 768 bits, d of 256 bits, and e of 880 bits.

The remainder of this paper is organized as follows. In section 2, we review some well-known attacks on the use of short secret exponent. In section 3, we propose and analyze a construction of RSA system to combat those short secret exponent attacks. In section 4, we present a variant of RSA such that the length of the secret exponent and the public exponent can be balanced and minimized. In section 5, the trade-off between the length of the secret exponent and the length of the public exponent is analyzed. Finally, we conclude this paper in section 6.

2 Overview of Previous Works

Because the security analysis of our schemes is related to Wiener's attack [26], Verheul and van Tilborg's attack [25], and Boneh and Durfee's attack [1] on the use of short secret exponent, here we briefly review these attacks as the background

information for reading this paper. Additionally, we also introduce the basic concept of unbalanced RSA which was proposed by Shamir [19].

2.1 Wiener's Attack and its Extension on Short Secret Exponent

Wiener's attack [26] is based on approximations using continued fractions to find the numerator and denominator of a fraction in polynomial time when a close enough estimate of the fraction is known. He showed the RSA system can be totally broken if the secret exponent with up to approximately one-quarter as many bits as the modulus (both p and q are of the same size). For simplicity, we slightly modify Wiener's attack in the following. Let $ed = k\phi(N)+1$ in a typical RSA system. Hence $\gcd(d, k)=1$. We can rewrite this equation as: $ed=k(N-(p+q)+1)+1$. Therefore, $|\frac{e}{N}-\frac{k}{d}|=\delta$, where $\delta=\frac{k}{d}\frac{p+q-1-\frac{1}{k}}{N}$. It is known that for a rational number x such that $|x-\frac{B}{A}|<\frac{1}{2A^2}$, where $\gcd(A, B)=1$, $\frac{B}{A}$ can be obtained as convergents of the continued fraction expansion of x. For further discussion of continued fractions, we refer the reader to [26]. As pointed out by Pinch [15], if $p<q<2p$ and $d<\frac{1}{3}N^{0.25}$, then $p+q-1<3\sqrt{N}$ and $k<d<\frac{1}{3}N^{0.25}$. Therefore, $|\frac{e}{N}-\frac{k}{d}|\le\frac{1}{dN^{0.25}}<\frac{1}{3d^2}<\frac{1}{2d^2}$. Thus $\frac{k}{d}$ can be found because $\frac{k}{d}$ is one of the $\log N$ convergents of the continued fraction for $\frac{e}{N}$.

The extension of Wiener's attack, proposed by Verheul and van Tilborg [25], basically follows Wiener's approach except that they proposed a more general method to compute the convergents of the continued fraction expansion of the same number as in Wiener's attack up to the point where the denominator of the convergent exceeds approximately $N^{0.25}$. For $d>N^{0.25}$, their attack need do an exhaustive search for about $2t+8$ bits, where $t=\log_2(d/N^{0.25})$. Because Verheul and van Tilborg's attack is not directly related to our work, we omit reviewing their attack here.

2.2 Boneh and Durfee's Attack on Short Secret Exponent

Based on solving the small inverse problem, Boneh and Durfee [1] proposed a new attack on short RSA secret exponent, which leads to a tighter bound than the bound proposed by Wiener. They concluded that if $e\approx N$ and $d<N^{0.292}$, then the secret exponent d can be efficiently found. In a typical RSA system, $ed = k\phi(N)+1$. So, $ed = k((N+1)-(p+q))+1$. Let $A=N+1$ and $s=-(p+q)$, and $t=-k$. Then $ed + t(A+s)=1$. Thus $t(A+s)=1 \pmod e$. If both t and s are much smaller than e, the problem can be viewed as follows: given an integer A, find an element close to A whose inverse modulo e is

small. This problem is usually referred as the *small inverse problem*. Let $e \approx N^{\alpha}$ and $d < N^{\beta}$. So far, Boneh and Durfee have showed that if $\beta < \frac{7}{6} - \frac{1}{3}(1+6\alpha)^{1/2}$, then the small inverse problem can be solved. Consequently, RSA is insecure whenever $d < N^{0.285}$ (which can be slightly improved up to $N^{0.292}$) if α=1.

2.3 Unbalanced RSA System

It is generally accepted that RSA moduli are composed of two large primes of the same size. Shamir [19] proposed a variant of the typical RSA, called *unbalanced RSA*, that the two primes are widely differing in size, e.g., $\log_2 q = 10 \cdot \log_2 p$. His motivation is to provide higher security without increasing computational cost.

In general, all the existing factoring algorithms to break RSA can be classified into two types: algorithms whose running time depends on the smaller factor p, and algorithms whose running time depends on the size of the modulus N. The fastest factoring algorithm of the first type is based on *elliptic curves*, and its asymptotic running time is $\exp(O((\log_2 p)^{1/2}(\log_2 \log_2 p)^{1/2}))$. This algorithm is usually referred as the *elliptic curve method* (ECM). So far, the largest factor that has ever been found in practice with this algorithm is about 53 digits (≈176 bits) long [6]. Therefore, if we choose p to be larger than 256 bits, the elliptic curve method becomes infeasible. The fastest factoring algorithm of the second type is based on the *general number field sieve* (GNFS), and its asymptotic running time is $\exp(O(\log_2 N)^{1/3}(\log_2 \log_2 N)^{2/3}))$. So far, the largest RSA modulus has ever been factored in practice with this algorithm is 140 digits (≈ 465 bits) long [5]. As the fast development of computer techniques and factoring algorithms, it is clear that the standard 512-bit RSA modulus no longer provides adequate security and must be significantly increased. Generally, for a large RSA modulus the GNFS attack is much more efficient than the ECM attack. Therefore, there is no need to increase the sizes of the RSA modulus and its prime factors at the same rate. Note that at the Eurocrypt'99 rump session, Shamir [20] announced his design for a special hardware, called "TWINKLE" device which can execute sieve-based factoring algorithms approximately two to three orders of magnitude as fast as a conventional fast PC. If the device can be implemented efficiently, this new technique will increase the size of factorable numbers by 100 to 200 bits for a GNFS attack.

Note that Gilbert *et al.* [7] pointed out that Shamir's unbalanced RSA suffers from some weaknesses. However, these weaknesses come from decrypting only modulo p (and thus limiting the plaintexts to integers smaller than p). Our schemes proposed in this paper don't suffer from the same weaknesses.

Note that some fast and practical public-key cryptosystems [11,14,23] which rely on the difficulty of factoring numbers of the type $p^2 q$ were proposed recently. These cryptosystems also use the same concept of making the factors short but large enough such that an ECM attack is infeasible.

3 RSA with Short Secret Exponent

In this section, we propose an unbalanced RSA system such that the use of short secret exponent in the RSA is still secure against all the existing short secret exponent attacks. We show that when p and q are the size of 256 bits and 768 bits, d of 192 bits is large enough to combat all the existing short secret exponent attacks.

3.1 The Proposed Scheme (Scheme I)

We propose a construction of the unbalanced RSA as follows:

Step 1. Randomly select a prime p and a prime q ($p<q$) such that p and N is large enough to make an ECM attack and a GNFS attack infeasible respectively, e.g., p and q are 256 bits and 768 bits long, and therefore N is about 1024 bits long.

Step 2. Randomly select a short secret exponent d such that $\log_2 d + \log_2 p > \frac{1}{3}\log_2 N$ (see Section 3.4) and $d > 2^{\gamma} p^{0.5}$, where γ is a security parameter, e.g., $\gamma = 64$ and hence d is 192 bits long. Note that it is necessary that γ satisfies the following inequality: $32\alpha\gamma\log_e 2 >> 3(1-\alpha-2\gamma\log_e 2)^2$, where $\alpha \approx \log_e q$. Here $\log_e$ denotes the logarithm with the base e that is the public exponent. We give the details in Section 3.3.

Step 3. Find e such that $ed=1 \bmod \phi(N)$, where $\phi(N)=(p-1)(q-1)$. Generally, e will be the same order of magnitude as $\phi(N)$. Here we assume $e \geq \phi(N)/2 + 1$ (the occurrence probability of this case is 1/2). If not, we repeat Step 2 again.

It is clear that the construction leads to a short secret exponent, e.g., a 192-bit d for a 1024-bit RSA modulus, which is far below the lower bounds proposed by Wiener (256 bits [26]) and by Boneh and Durfee (299 bits [1]). Note that if $\gamma \approx 0.5\log_2 p$, to our best knowledge, no information can be obtained to break the resulting RSA system until now. The details are explained in Section 3.3. So, the RSA system with p of 256 bits, q of 768 bits, d of 256 bits (due to 128-bit γ) and e of 1024 bits is quite secure.

3.2 Combating Wiener's Attack and its Extension

Because $d > 2^{\gamma} p^{0.5}$, it is clear that $\frac{1}{p} > 2^{2\gamma}\frac{1}{d^2}$. Because $k\phi(N)=ed-1$, we can obtain $\frac{k}{d} = \frac{e-\frac{1}{d}}{\phi(N)} \geq \frac{\frac{\phi(N)}{2}+1-\frac{1}{d}}{\phi(N)} \geq \frac{\frac{\phi(N)}{2}}{\phi(N)} \geq \frac{1}{2}$. Thus $|\frac{e}{N}-\frac{k}{d}| = \frac{k}{d}\frac{p+q-1-\frac{1}{k}}{N} > \frac{k}{d}\frac{q}{N} = \frac{k}{d}\frac{1}{p} > \frac{1}{2}\,2^{2\gamma}\frac{1}{d^2} = 2^{2\gamma}\frac{1}{2d^2} >> \frac{1}{2d^2}$. If γ is adequately large, the value $|\frac{e}{N}-\frac{k}{d}|$ will be far away the value $\frac{1}{2d^2}$. Thus, Wiener's attack doesn't apply to Scheme I.

3.3 Combating Boneh and Durfee's Attack

Following Boneh and Durfee's approach, let $A=N+1$, $s=-(p+q)$, and $t=-k$. Thus $t(A+s) \equiv 1 \pmod{e}$. Let $|s|<e^{\alpha}$ and $|t|<e^{\beta}$. The sufficient condition for solving the small inverse problem is: $4\alpha(2\beta+\alpha-1) < 3(1-\beta-\alpha)^2$. Because of the limit of space, we provide the details in the full version of this paper.

In our construction $e \approx N$, $q \approx |s| \approx e^{\alpha}$, $d \approx |k| \approx |t| \approx e^{\beta}$, therefore $p=\frac{N}{q} \approx \frac{e}{e^{\alpha}} \approx e^{1-\alpha}$. Hence $d \approx 2^{\gamma} p^{0.5} \approx 2^{\gamma} e^{0.5(1-\alpha)}$. Let $2^{\gamma} \approx e^{\gamma'}$, i.e., $\gamma' \approx \gamma \log_e 2$. Therefore, $d \approx e^{\gamma'+0.5(1-\alpha)}$. Thus, $2\beta \approx 2\gamma'+1-\alpha$. So, the sufficient condition for solving the small inverse problem can be reduced into $32\alpha\gamma' < 3(1-\alpha-2\gamma')^2$. Hence, in order to combat Boneh and Durfee's attack, it is necessary that γ is adequately large such that the following inequality holds: $32\alpha\gamma \log_e 2 >> 3(1-\alpha-2\gamma \log_e 2)^2$. As an example, we assume p, q, γ and d are 256 bits, 768 bits, 64 bits, and 192 bits long respectively. Thus α=0.75 and β=0.1875. It is clear that $4\alpha(2\beta+\alpha-1) = 0.375 >> 3(1-\beta-\alpha)^2 = 0.117186$. So, Boneh and Durfee's attack cannot succeed.

An important observation proposed by Boneh and Durfee [1] is that the unique solution of the small inverse problem encodes enough information to find d. Therefore, a strong resistance to Boneh and Durfee's attack is to make the small inverse problem failed to have a unique solution. This is why Boneh and Durfee believed that a typical RSA with $d \approx N^{0.5}$ is strongly secure against short secret exponent attacks. So, if we let γ be slightly larger than $0.5\log_2 p$, then $d > p$. Without loss of generality, we assume $|t| \approx d > p \approx e^{1-\alpha}$, then $|t| > e^{1-\alpha}$. Thus the resulting small inverse problem: $t(A+s) \equiv 1 \pmod{e}$, where $|t| > e^{1-\alpha}$ and $|s| \approx e^{\alpha}$, will no longer have a unique solution. As a result, if d is a few bits larger than p, the

resulting RSA is strongly secure against Boneh and Durfee's attack even if Boneh and Durfee's attack can be up to $d< N^{0.5}$.

3.4 Combating the Cubic Attack

Here we consider a kind of attack, named the cubic attack, in the following.
Because $ed=k(p-1)(q-1)+1$ and $N=pq$, we can obtain the following system of modular equations:
(1) $k(p-1)(q-1)+1 = 0 \pmod e$
(2) $pq = N \pmod e$.
Combining (1) and (2), we can obtain the following cubic equation in two variables k and p:
(3) $k(p-1)(N-p)+p = 0 \pmod e$
Coppersmith [2] has shown how to solve such cubic equations heuristically if $\log_2 k + \log_2 p < \frac{1}{3}\log_2 e$. To combat Coppersmith's attack, we need the constraint: $\log_2 d + \log_2 p > \frac{1}{3}\log_2 N$ because $\log_2 k \approx \log_2 d$ and $\log_2 e \approx \log_2 N$ in Scheme I. On the other hand, if one can know the exact value of k, then the equation (3) can be reduced to a quadratic equation in a single variable p and hence can be solved provided that either e is prime, or can be factored and doesn't have too many prime factors. Therefore, we must make k unknown to an attacker. In Scheme I, because k is of the same order of magnitude as d, it is large enough to make an exhaustive search infeasible. Hence Scheme I is secure against the cubic attack.

4 RSA with Balanced Public Exponent and Secret Exponent

Traditionally, when constructing RSA, p and q are first selected. After that, either first select the secret exponent d and then determine the public exponent e, or vice versa. Thus either e or d is of the same order of magnitude as $\phi(N)$. In this section, we are interested in constructing RSA with balanced and minimized public and secret exponents such that both are approximately $(\frac{1}{2}\log_2 N + 56)$ bits long without compromising the security of RSA. Different from traditional constructions, we first select p and d, and then determine e and q.

4.1 The Proposed Scheme (Scheme II)

Theorem 1. Let two integers $a, b > 1$. If $\gcd(a, b) =1$, then we can find a unique pair (u_h, v_h) satisfying $au_h-bv_h=1$, where $(h-1)b< u_h < hb$ and $(h-1)a< v_h< ha$, for any integer $h\geq 1$.

We assume p and q are approximately $(\frac{1}{2}\log_2 N - 112)$ and $(\frac{1}{2}\log_2 N + 112)$ bits long respectively. Here we assume that p and N are large enough to make an ECM attack and a GNFS attack infeasible, e.g., p and N are about 400 bits and 1024 bits long respectively. Our construction is the following:

Step 1. Randomly select a prime number p of $(\frac{1}{2}\log_2 N - 112)$ bits.
Step 2. Randomly select a number k of 112 bits.
Step 3. Randomly select a number d of $(\frac{1}{2}\log_2 N + 56)$ bits such that $gcd(k(p-1), d)=1$.
Step 4. Based on Theorem 1, we can uniquely determine two numbers u' and v' such that $du'-k(p-1)v'=1$, where $0< u' < k(p-1)$ and $0< v'< d$.
Step 5. If $gcd(v'+1, d) \neq 1$, then go to Step 3.
Step 6. Randomly select a number h of 56 bits, compute $u = u'+hk(p-1)$ and $v = v'+hd$.
Step 7. If $v+1$ isn't a prime number, then go to Step 6.
Step 8. Let $e=u$, $q=v+1$, and $N=pq$, then p, q, e, d, and N are the parameters of RSA.

Clearly, in this construction e and d satisfy the equation: $ed = k(p-1)(q-1)+1=k\phi(N)+1$. Therefore, the equation: $ed=1 \bmod \phi(N)$ still holds as that in typical RSA. Obviously, both e and d obtained from this construction are approximately $(\frac{1}{2}\log_2 N + 56)$ bits long, and p and q are approximately $(\frac{1}{2}\log_2 N - 112)$ bits and $(\frac{1}{2}\log_2 N + 112)$ bits long respectively. As an example, if $\log_2 N \approx 1024$, then d is 568 bits long, p is 400 bits long, e is about 568 bits long and q is about 624 long. A concrete example for this case is given in **Appendix A**. In order to measure the efficiency of the proposed scheme, we ran some experiments to test the average times required to find a suitable h in Step 6 for obtaining a prime q. Under 100 samples, our results indicate that in average we need try 487.48 times for Step 6 when N is of 1024 bits long. A comparative result is 566.31 times of selecting a random number of 624 bits and testing whether the number is a prime. This shows that both have approximately the same cost in order to obtain a prime q. Note that in Step 5, if $gcd(v'+1, d) \neq 1$, it implies that it is impossible to find h such that $v'+hd+1$ is a prime. In addition, the prime p generated in Step 1 can be arbitrarily determined, e.g., selecting a strong prime p, but the prime q generated in Step 8 cannot. Fortunately, the requirement for RSA key that p and q are strong primes is no longer needed due to [17,21-22].

Note that compared with the RSA with CRT-based implementations, Scheme II apparently doesn't provide better efficiency. However the CRT-based RSA needs to keep more secrets p and q than the typical RSA. Moreover, the CRT-based RSA usually incurs some additional security problems [12], even some error detection techniques are applied to it.

4.2 Combating Wiener's Attack and its Extension:

Here we examine the security of Scheme II following the line of the attack, proposed by Wiener, on short RSA secret exponent.

It is clear that $|\frac{e}{N}-\frac{k}{d}|=\frac{k}{d}\frac{p+q-1-\frac{1}{k}}{N} > \frac{k}{d}\frac{q}{N}=\frac{k}{d}\frac{1}{p}$. Without loss of generality, we assume that $k> 2^{111}$, $2^{-113}N^{0.5}<p<2^{-112}N^{0.5}$ and $2^{55}N^{0.5}<d<2^{56}N^{0.5}$. Hence, $\frac{1}{p}>\frac{2^{112}}{N^{0.5}}$ and $\frac{1}{N^{0.5}}>2^{55}\frac{1}{d}$. Thus $\frac{k}{d}\frac{1}{p}> 2^{111}\frac{1}{d}\frac{2^{112}}{N^{0.5}}>2^{279}\frac{1}{2d^2}>>\frac{1}{2d^2}$. So, $|\frac{e}{N}-\frac{k}{d}|$ will be much larger than $\frac{1}{2d^2}$. Thus, Wiener's attack doesn't apply to Scheme II.

4.3 Combating Boneh and Durfee's Attack:

Similar to Section 3.3, the sufficient condition for solving the small inverse problem is: $4\alpha(2\beta+\alpha-1)<3(1-\beta-\alpha)^2$. Due to the difficulty of obtaining a general proof, we only show Scheme II with N of 1024 bits is secure against Boneh and Durfee's Attack. For Scheme II, if $\log_2 N\approx 1024$, then d is 568 bits long, p is 400 bits long, e is about 568 bits long and q is about 624 bits long. Thus $\alpha=\frac{624}{568}$ and $\beta=\frac{112}{568}$. It is clear that $4\alpha(2\beta+\alpha-1)= 2.1664 >> 3(1-\beta-\alpha)^2=0.2625$. So, Boneh and Durfee's attack doesn't apply to Scheme II.

4.4 Combating the Cubic Attack

Here we refer to Section 3.4. In Scheme II, because $\log_2 k+\log_2 p=\frac{1}{2}\log_2 N >> \frac{1}{3}\log_2 e$, Coppersmith's attack cannot work here. In addition, because k is 112 bits long, it is large enough to make an exhaustive search infeasible. Hence Scheme II is secure against the cubic attack.

5 Trade-off between Public Exponent and Secret Exponent

From Section 4, we know that it is possible for us to use median public and secret exponents in RSA system such that the overall computations required in encryption and decryption are minimized and balanced without compromising with the security of RSA. Therefore, one may be desirable to have secret and public exponents which are differing in size, but the overall computations are still minimized, e.g., $d \approx N^{0.25}$ and $e \approx N^{0.86}$. To minimize the overall computations required in encryption and decryption, it is natural that there exists a trade-off between the length of the public exponent and the length of the secret exponent. In this section, we are interested in addressing this problem.

5.1 The Proposed Scheme (Scheme III)

In the following, generalizing Scheme II, we give an efficient construction of RSA such that $\log_2 e + \log_2 d \approx \log_2 N + l_k$, where l_k is a predetermined constant.

Step 1. Randomly select a prime number p of length l_p ($l_p < \frac{1}{2}\log_2 N$) such that it is large enough to make an ECM attack infeasible, e.g., l_p=256.

Step 2. Randomly select a number k of length l_k, e.g., l_k=112.

Step 3. Randomly select a number d of length l_d such that $gcd(k(p\text{-}1), d)$=1, e.g., l_d=256.

Step 4. Based on Theorem 1, we can uniquely determine two numbers u' and v' such that $du'\text{-}k(p\text{-}1)v'$=1, where $0< u' < k(p\text{-}1)$ and $0< v' < d$.

Step 5. If $gcd(v'+1, d) \neq 1$, then go to Step 3.

Step 6. Randomly select a number h of length $\log N - l_p - l_d$, compute $u = u'+hk(p\text{-}1)$ and $v = v'+hd$.

Step 7. If v+1 isn't a prime number, then go to Step 6.

Step 8. Let $e=u$, $q=v+1$, and $N=pq$, then p, q, e, d, and N are the parameters of RSA.

From Step 1-3, we know that k, p, and d are l_k bits, l_p bits, and l_d bits long. Obviously, e and q obtained from the above construction are roughly $\log N + l_k - l_d$ bits and $\log_2 N - l_p$ bits long. These parameters l_k, l_p, and l_d must satisfy the following requirements:

(1) $l_k >> l_p - l_d + 1$. (See 5.2)

(2) α and β must satisfy: $4\alpha(2\beta + \alpha - 1) >> 3(1 - \beta - \alpha)^2$, where $\alpha = \frac{\log_2 N - l_p}{\log_2 N + l_k - l_d}$ and $\beta = \frac{l_k}{\log_2 N + l_k - l_d}$. (See 5.3)

(3) k is large enough to make an exhaustive search infeasible and $l_k + l_p > \frac{1}{3}\log_2 N$ (see Section 5.4)

As an example, if k, p, and d are 112 bits, 256 bits, and 256 bits long, then e is about 880 bits long and q is about 768 bits long. A concrete example for this case is given in **Appendix B**. In order to measure the efficiency of the proposed scheme, we also ran some experiments to test the average times required to find a suitable h in Step 6 for obtaining a prime q. Under 100 samples, our results indicate that in average we need try 743.56 times for Step 6. A comparative result is 696.86 times of selecting a random number of 768 bits and testing whether the number is a prime. This shows that both have approximately the same cost in order to obtain a prime q.

Note that if one wish to have a smaller public exponent and a larger secret exponent, he need only modify this construction by interchanging the positions of e and d, i.e., he first fixes e and p and then determines d and q.

5.2 Combating Wiener's Attack and its Extension

Here we refer to Section 3.2. It is clear that $|\frac{e}{N} - \frac{k}{d}| > \frac{k}{d}\frac{1}{p}$. Without loss of generality, we assume that $k > 2^{l_k - 1}$, $2^{l_p - 1} < p < 2^{l_p}$ and $2^{l_d - 1} < d < 2^{l_d}$. Hence, $\frac{1}{p} > 2^{-l_p}$ and $2^{-l_d + 1} > \frac{1}{d}$. Obviously, $\frac{k}{d}\frac{1}{p} > 2^{l_k - l_p - 1}\frac{1}{d} = 2^{l_k - l_p}\frac{1}{2d}$. From requirement (1): $l_k >> l_p - l_d + 1$, we know that $l_k - l_p >> -l_d + 1$. Therefore, $\frac{k}{d}\frac{1}{p} >> 2^{-l_d + 1}\frac{1}{2d} > \frac{1}{2d^2}$. So, $|\frac{e}{N} - \frac{k}{d}|$ is much larger than $\frac{1}{2d^2}$. Thus, Wiener's attack doesn't apply to Scheme III.

5.3 Combating Boneh and Durfee's Attack

Because k, p, and d are l_k bits, l_p bits, and l_d bits long, e and q obtained from Scheme III will be roughly $\log N + l_k - l_d$ bits and $\log N - l_p$ bits long. Note that $q > p$. Let $|s| < e^{\alpha}$ and $|t| < e^{\beta}$. Therefore, $\alpha \approx \frac{\log_2 N - l_p}{\log_2 N + l_k - l_d}$ and $\beta \approx \frac{l_k}{\log_2 N + l_k - l_d}$. As described in Section 3.3, to combat Boneh and Durfee's attack, α and β must satisfy: $4\alpha(2\beta + \alpha - 1) >> 3(1 - \beta - \alpha)^2$. As an example, if k, p, and d are 112 bits, 256 bits, and 256 bits long (hence e and q are about 880 bits and

768 bits long), then $\alpha \approx \frac{768}{880}$ and $\beta \approx \frac{112}{880}$. It is clear that $4\alpha(2\beta+\alpha-1) = 0.4443 >> 3(1-\beta-\alpha)^2 = 0$.

5.4 Combating the Cubic Attack

Here we refer to Section 3.4. In Scheme III, because $\log_2 k + \log_2 p > \frac{1}{3}\log_2 N > \frac{1}{3}\log_2 e$, Coppersmith's attack cannot work here. Besides, l_k=112 makes an exhaustive search infeasible. Therefore, Scheme III is secure against the cubic attack.

6 Conclusions

An important observation obtained in this paper is that making the size of *p* and *q* different enhances RSA to combat all the existing short secret exponent attacks. Although this also reduces the strength of RSA against factoring, *p* of 256 bits is large enough to combat an ECM attack at present. Hence RSA with *d* of 192 bits, *p* of 256 bits, and *q* of 768 bits is secure against Boneh and Durfee's attack. To our best knowledge, *d* of 256 bits is quite secure even if Boneh and Durfee's attack can be up to d< $N^{0.5}$. We also propose an efficient construction of RSA with $\log_2 e = \log_2 d = \log_2 N$+56 and its generalization with $\log_2 e + \log_2 d \approx \log_2 N + l_k$ where l_k is a predetermined constant. These two constructions are also secure against all the existing short secret exponent attacks due to making the size of *p* and *q* different. As an example, RSA with *e* of 568 bits, *d* of 568 bits, *p* of 400 bits, and *q* of 624 bits and RSA with *p* of 256 bits, *q* of 768 bits, *d* of 256 bits, and *e* of 880 bits are both secure.

Remark: After we finished this paper, Marc Joye provided us with a related article by Sakai, Morii, and Kasahara [18]. In the paper, they proposed a key generation algorithm for RSA cryptosystem which can make $\log_2 e + \log_2 d \approx \log_2 N$. Their schemes have the following properties:

(1) $ed = \frac{k(p-1)(q-1)}{2g} + 1$, where *g* is a large prime and *g*|(*p*-1), *g*|(*q*-1)

(2) $\log_2 p \approx \log_2 q \approx \frac{1}{2}\log_2 N$

It should be noticed that Wiener [26] has pointed out that making g large (and hence GCD(*p*-1, *q*-1) is large) may cause some security problems. For example, one can find *g* from *N*-1 by factoring algorithms because *g* divides *pq*-1=(*p*-1)(*q*-1)+(*p*-1)+(*q*-1). If *g* isn't large enough to combat an ECM attack, e.g., *g* of 110 bits and 120 bits in Sakai et al.'s schemes, *g* can be found from *N*-1 easily. Even if *g* is large enough to

combat an ECM attack, e.g, 250 bits long, it is still possible to factor N-1 and hence obtain g because N-1 may possibly contain only some small prime factors excluding g. Once g is obtained, Wiener's attack can work efficiently [26]. A possible solution to repair their schemes is to make g much larger (> 250 bits) and let N-1 contain at least two large prime factors (> 250 bits) including g. However, in some literatures and current practical use, e.g., X9.31, it is usually recommended to make GCD(p-1, q-1) small in order to guard against the relevant attacks such as repeat encryption attacks.

Acknowledgments. We are grateful to Marc Joye for providing us with reference [18] and his valuable comments. We also thank Sung-Ming Yen and the anonymous referees for their helpful comments.

References

1. D. Boneh and G. Durfee, "Cryptanalysis of RSA with private exponent $d < N^{0.292}$", *Proc. of EUROCRYPT'99*, LNCS 1592, Springer-Verlag, pp. 1-23, 1999.
2. D. Coppersmith, "Finding a small root of a univariate modular equation", *Proc. of EUROCRYPT'96*, LNCS 1070, Springer-Verlag, pp. 155-165, 1996.
3. D. Coppersmith, "Small solutions to polynomial equations, and low exponent RSA vulnerabilities", *Journal of Cryptology*, Vol. 10, pp. 233-260, 1997.
4. D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter, "Low-exponent RSA with related messages", *Proc. of EUROCRYPT'96*, LNCS 1070, Springer-Verlag, pp. 1-9, 1996.
5. S. Cavallar, W. Lioen, H. te Riele, B. Dodson, A. Lenstra, P. Leyland, P.L. Montgomery, B. Murphy, P. Zimmermann, "Factorization of RSA-140 using the Number Field Sieve", Proc. of ASIACRYPT'99, Springer-Verlag, 1999.
6. ECMNET Project; http://www.loria.fr/~zimmerma/records/ecmnet.html
7. H. Gilbert, D. Gupta, A. Odlyzko, and J.J. Quisquater, "Attacks on Shamir's RSA for paranoids", *Information Processing Letters*, Vol. 68, pp. 197-199, 1998.
8. J. Hastad, "On using RSA with low exponent in a public key network", *Proc. of CRYPTO'85*, LNCS, Springer-Verlag, pp. 403-408, 1986.
9. J. Hastad, "Solving simultaneous modular equations of low degree", *SIAM J. of Computing*, Vol. 17, pp. 336-341, 1988.
10. I.N. Herstein, *Topics in Algebra*, Xerox Corporation, 1975.
11. D. Hühnlein, M.J. Jacobson, S. Paulus, and T. Takagi, "A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption", *Proc. of EUROCRYPT'98*, LNCS 1403, Springer-Verlag, pp. 294-307, 1998.
12. M. Joye, J.J. Quisquater, S.M. Yen, and M. Yung, "Security paradoxes: how improving a cryptosystem may weaken it", *Proceedings of the Ninth National Conference on Information Security*, Taiwan, pp. 27-32, May 14-15, 1999.
13. A. Lenstra, "Generating RSA moduli with a predetermined portion", *Proc. of ASIACRYPT'98*, LNCS 1514, Springer-Verlag, pp. 1-10, 1998.
14. T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring", *Proc. of EUROCRYPT'98*, LNCS 1403, Springer-Verlag, pp. 308-318, 1998.
15. R. Pinch, "Extending the Wiener attack to RSA-type cryptosystems", *Electronics Letters*, Vol. 31, No. 20, pp. 1736-1738, 1995
16. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communication of the ACM*, Vol. 21, pp. 120-126, 1978.

17.R. Rivest and R. D. Silverman, “Are strong primes needed for RSA?”, in The 1997 RSA Laboratories Seminar series, Seminar Proceedings, 1997.
18.R. Sakai, M. Morii, and M. Kasahara, “New key generation algorithm for RSA cryptosystem”, *IEICE Trans. Fundamentals*, Vol. E77-A, No. 1, pp. 89-97, 1994
19.A. Shamir, “RSA for paranoids”, *CryptoBytes*, Vol. 1, No. 3, pp. 1,3-4, 1995.
20.A. Shamir, “Factoring large numbers with the TWINKLE device”, presented at *Eurocrypt'99*, 1999.
21.R. D. Silverman, “Fast generation of random, strong RSA primes”, *CryptoBytes*, Vol. 3, No. 1, pp. 9-13, 1997.
22.R. D. Silverman, “The requirement for strong primes in RSA”, RSA Laboratories Technical Note, May 17, 1997.
23.T. Takagi, “Fast RSA-type cryptosystem modulo p^2q”, *Proc. of CRYPTO’98*, LNCS 1462, Springer-Verlag, pp. 318-326, 1998.
24.S.A. Vanstone and R.J. Zuccherato, “Short RSA keys and their generation”, *Journal of Cryptology*, Vol. 8, pp. 101-114, 1995.
25.E. Verheul and H. van Tilborg, “Cryptanalysis of less short RSA secret exponents”, *Applicable Algebra in Engineering, Communication and Computing*, Springer-Verlag, Vol. 8, pp. 425-435, 1997.
26.M. Wiener, “Cryptanalysis of short RSA secret exponents”, *IEEE Transactions on Information Theory*, Vol. 36, No. 3, pp. 553-558, 1990.

Appendix A: An example for *p* of 400 bits, *q* of 624 bits, *d* of 568 bits, and *e* of 568 bits

p=0000cd0a 73cb74b6 27aa29e7 9b1a3c1b d73f4b67 92abde25 c2dcc2dd 68f7a477 9cc6f0a0 d5eeea7c 7c740c8c b370a2e1 6112a393
q=0000807e 4aac7213 62d7d547 4e4dac07 1ea03096 0f13c597 a619a6d7 4c8a3e5b dcd00bcb dcfb0758 555f6b4e 23cc4f6a 5221fa87 bfef172d b815a296 4c5c5be7 61a22fe4 53808fac 0a2fb2d2 548285af
d=008dc2d0 0c1e3027 e0a43f18 022896a0 35379c76 b1e5577c 71038464 bf9ef9a6 00bb3aa0 bb4f590d ef8311ab 95282426 7277f349 200c5d67 5e23dc05 9613dccc ae0a5dad 1209cc53
e=00b335b3 9edd0f90 546f4a51 2ec2a0dd 191e1fb0 38f6b5dd b9ef5156 7ecdc538 355a67b6 d7fbbee3 0926925c b0112914 bbe9f4bf a1a61f92 53dfab7e d9c40261 6fc3d7a8 f77c025f

Appendix B: An example for *p* of 256 bits, *q* of 768 bits, *d* of 256 bits, and *e* of 880 bits

p=f80dd4da c85afbd9 019d0f24 92c03006 c5baef83 7cfc15eb 2e17b1c1 1fb166e3
q=96d12784 058456cf 00e17f03 b6402825 00a95a1a 772f7059 ea78ac03 57e49dbf feaff1d1 b556e47f 855e8d74 9905753b 12a46068 ce6df746 0e85602c 8f4ed8ac ed6b7f21 2fb1d58f ca645447 ae39277d d01e681a e8a630c6 8c158859 c2e4b743
d=bd82175c 6d9bd203 9ce3f83b cdbceb8e 51c82b29 7f4e237d b0eb3518 807c02bf
e=0000c3b8 1c856425 ff98f54d 605ebe3e 58fd6381 acd328b8 0c4c1d7d ebba6832 061d6fa7 baa8b814 65a82be5 93cdc56a 21ac87e7 693e97e9 3632dfc7 47572a58 f3683163 cd312935 bd24a7ac 08204830 1ba73867 da7456d7 f5efcada 715ad9a0 cec3edd3 e773421b 2c699c42 ef62ebff