On the Design of Rebalanced RSA-CRT

Hung-Min Sun[1], M. Jason Hinek[2], Mu-En Wu[3]

[1, 3]Department of Computer Science
National Tsing Hua University, Hsinchu, Taiwan 300
E-mail: [1]hmsun@cs.nthu.edu.tw; [3]mn@is.cs.nthu.edu.tw

[2]School of Computer Science
University of Waterloo, Waterloo, Ontario, N2L-3G1, Canada
E-mail: mjhinek@alumni.uwaterloo.ca

October 5, 2005

Abstract

In 1982, Quisquater & Couvreur proposed a variant of RSA based on the Chinese Remainder Theorem, called RSA-CRT, to speed up RSA decryption. In 1990, Wiener suggested another variant, called Rebalanced RSA-CRT, which further speeds up RSA decryption by shifting decryption costs to encryption costs. However, this approach essentially maximizes the encryption time since the public exponent e in Rebalanced RSA-CRT is generally about the same order of magnitude as the RSA modulus. In this work[1], we introduce two variants of Rebalanced RSA-CRT in which the public exponent e is much smaller than the modulus, thus reducing the encryption costs, while maintaining low decryption costs. For a 1024-bit modulus, our first variant (Scheme A) offers encryption times that are at least 2.6 times faster than that in the original Rebalanced RSA-CRT, while the second variant (Scheme B) offers encryption times at least 3 times faster. In both variants, the decrease in encryption costs is obtained at the expense of slightly increased decryption costs and increased key generation costs. Thus, the variants proposed here are best suited for applications which require low encryption/decryption costs but do not require key generation on the fly.

Index Terms --- RSA, RSA-CRT, CRT, Rebalanced RSA-CRT, Lattice Basis Reduction.

1. Introduction

Many practical issues have been considered when implementing RSA such as how to reduce the storage requirement for the RSA modulus, how to reduce the encryption time (or signature-verification time), how to reduce the decryption time (or signature-generation time) [5], how to balance the encryption and decryption time [23, 24], and so on. In this work we are concerned with minimizing the overall cost of encryption and decryption combined (or the cost of signature generation and verification combined).

Encryption and decryption in RSA each require an exponentiation modulo a large integer N which is the product of two large primes, p and q. The exponent in the exponentiations is the public exponent e for encryption and the private exponent d for decryption. Using the standard double-and-square method for modular exponentiation, which is commonly used, the cost for encryption and decryption is roughly proportional to the number of bits in e and d, respectively.

To simply reduce the encryption time (or the signature-verification time), one can use a small public exponent e. The smallest possible value for the public exponent is e=3, however a more widely accepted and used small public exponent is $e=2^{16}+1=65537$ [7, 8]. With an appropriate padding scheme, such as OAEP, using such small public exponents is considered safe. On the other hand, to simply reduce the decryption time (or the signature-generation time), one can use a small private exponent d. However, unlike the small public exponent scenario, using a very small private exponent is insecure [2]. Indeed, instances of RSA with private exponent $d<N^{0.25}$, can be efficiently broken with Wiener's continued fraction attack [27, 28] and Boneh & Durfee's lattice-based attack [3] shows that and instance of RSA with $d<N^{0.292}$ should be considered unsafe.

A more sophisticated way to reduce the decryption time is to employ the Chinese Remainder Theorem (CRT) [19]. Using this technique, two half-sized modular exponentiations are required. Consider an instance of RSA with modulus $N=pq$, public exponent e, and private exponent d which is roughly the same order of magnitude as N. To decrypt a given a ciphertext C using the standard definition of RSA, one would compute the plaintext as $M \equiv C^d \pmod{N}$. Using the CRT, one first computes $C_p \equiv C^{d_p} \pmod{p}$ and $C_q \equiv C^{d_q} \pmod{q}$, where $d_p \equiv d \pmod{p-1}$ and $d_q \equiv d \pmod{q-1}$, and then combines these with the CRT to obtain the final result $M \equiv C^d \pmod{N}$. Such an approach, called RSA-CRT, achieves decryption times that are 4 times faster than decryption using the standard RSA definition. And, since there are no restrictions on the public exponent e, we are free to use a very small public

[1] This is an improved version of [25, 26].

exponent to reduce the encryption costs.

One can further reduce the decryption time by carefully choosing d so that both $d_p \equiv d \pmod{p-1}$ and $d_q \equiv d \pmod{q-1}$ are small [28]. In the key generation phase one first selects two small CRT-exponents d_p and d_q, which are combined, via the CRT, to obtain the private exponent d satisfying $d_p \equiv d \pmod{p-1}$ and $d_q \equiv d \pmod{q-1}$. The public exponent is then simply the inverse of d modulo $\phi(N)$, where $\phi(N)$ is Euler's totient function. Such a variant of RSA-CRT, called Rebalanced RSA-CRT, enables us to rebalance the costs of encryption and decryption. In other words, we can speed up the CRT decryption by shifting the decryption cost to the encryption cost. Note that in Rebalanced RSA-CRT, both d and e will, with very high probability, be roughly the same order of magnitude as the modulus N. Since the decryption time depends on the bit-size of d_p and d_q, not on the bit-size of d, the decryption time is minimized. But, since the encryption time depends on the bit-size of e, encryption times for Rebalanced RSA-CRT will essentially be maximized. Whether or not there exist efficient attacks on Rebalanced RSA-CRT remains an open problem. So far, the best known attack on Rebalanced RSA-CRT runs in time complexity $O(\min\{d_p^{0.5}, d_q^{0.5}\})$ which is exponential in the bit-size of d_p or d_q [2, 20]. To defend against this attack, the bit-size of d_p and d_q should each be at least 160 bits to match the estimated complexity of factoring a 1024-bit RSA modulus. Using 160-bit CRT-exponents with a 1024-bit modulus, decryption in Rebalanced RSA-CRT will be about 3.2 (=512/160) times faster than that in RSA-CRT. Of course, since e is roughly the same order of magnitude as N, encryption in Rebalanced RSA-CRT is essentially maximized. If the public exponent e could be made to be much smaller than N, while keeping small CRT-exponents, then both encryption and decryption times could be reduced which would be more convenient and practical in many applications.

In this work, we are interested in studying the following problem: Do secure variants of Rebalanced RSA-CRT exist whose public exponent e is much shorter than N? We answer this by proposing two variants of Rebalanced RSA-CRT; Scheme A and Scheme B. Each scheme generates instances of Rebalanced RSA-CRT with public exponents much smaller than N. This is achieved by shifting some of the encryption costs to key generation costs and decryption costs.

This paper is a revised version of earlier results in [25, 26, 29], in which the parameters suggested are not secure. Also, it should be pointed out that Galbraith, Heneghan & McKee [13] proposed a similar but independent scheme in their work, Tunable Balancing of RSA, which will be discussed more in later sections.

The remainder of this paper is organized as follows. We begin with a brief review of RSA and some well-known variants of RSA in Section 2. Next we give an overview of the best known attacks against these variants in Section 3. In Section 4 we present our proposed variants, Scheme A and Scheme B, and analyze their security in Section 5. Some performance data and comparisons of our proposed schemes with the other variants of RSA are given in Section 6. Finally, we conclude with some final remarks in Section 7.

2. RSA and Some of its Fast Variants

In this section we review the original presentation of RSA along with some variants of it. In particular, we are interested in variants (or parameter choices) that reduce the encryption or decryption time. We do not, however, consider multi-prime or multi-power RSA in this work. For a survey on fast variants of RSA see Boneh & Shacham [5]. Even though the variants we discus below apply to both the RSA cryptosystem and RSA signature scheme, in order to simplify the exposition we will only consider the RSA cryptosystem. Thus, for each variant we will be concerned with three algorithms: key generation, encryption, and decryption. The security of each of the variants considered will be discussed in the next section. Besides, the notation introduced for each variant will remain the same for the rest of this work.

2.1. Original RSA

In the original version of the RSA cryptosystem [21], we have the following three algorithms:

Key Generation: Choose two distinct large random primes p and q. Their product, $N=pq$, is called the RSA modulus. Choose a large random integer d that is relatively prime to $\phi(N)=(p-1)\cdot(q-1)$. Finally, the integer e is computed as the multiplicative inverse of d modulo $\phi(N)$. The pair (e, N) is the public key and the pair (d, N) is the private key. The integer e is called the encryption (or public) exponent while the integer d is called the decryption (or private) exponent. Note that with high probability the public exponent e will be roughly the same order of magnitude as $\phi(N)$.

Since the public and private exponents are inverses of each other modulo $\phi(N)$, we know that $ed \equiv 1 \pmod{\phi(N)}$. We call this the key relation. Also, there must exist an integer k such that $ed=1+k\cdot\phi(N)$. We call this the key equation.

We call the RSA primes, p and q, balanced if they are roughly the same size. In particular, they are balanced if $0.5(N)^{0.5}<p<N^{0.5}<q<2N^{0.5}$ (where p and q can be interchanged). We will only consider instances of RSA with balanced primes. A consequence of this is that $\phi(N)=(p-1)\cdot(q-1)=N-\Lambda$, where $\Lambda<3N^{0.5}$. Thus, N is a good approximation of $\phi(N)$ when p and q are balanced.

Encryption: A plaintext message $M \in Z_N$ is encrypted by raising it to the e-th power modulo N. The result, $C \equiv M^e \pmod{N}$, is called the ciphertext of M. Notice that $C \in Z_N$ also. All variants discussed below use this algorithm for encryption.

Decryption: A ciphertext $C \in Z_N$, for a given plaintext message M, is decrypted by raising it to the d-th power modulo N. This follows since $C^d \pmod N \equiv M^{ed} \pmod N \equiv M^{1+k\cdot\phi(N)} \pmod N \equiv M \pmod N = M$.

An obvious way to decrease encryption or decryption time is to simply use a small public or private exponent. We consider each as a fast variant of RSA.

RSA-Small-*e*: We call instances of RSA with a small public exponent **RSA-Small-*e***. Encryption and decryption remain the same. Key generation is slightly modified though. For a random small public exponent of size n_e (ie, e with bit-length n_e), first choose the RSA primes then find a random n_e-bit integer e that is relatively prime to $\phi(N)$. The private exponent is simply computed as the inverse of e modulo $\phi(N)$. Note that d will be roughly the same order of magnitude as $\phi(N)$ with very high probability. For a fixed public exponent, $e=2^{16}+1$ for example, the primes are each randomly chosen so that they are relatively prime to the predetermined e. Again, $d \equiv e^{-1} \pmod{\phi(N)}$. The smallest possible value for a public exponent is $e=3$, but the value $e=2^{16}+1$ is the most commonly used fixed value [7, 8].

RSA-Small-*d*: We call instances of RSA with a small private exponent **RSA-Small-*d***. As with RSA-Small-e, encryption and decryption remain the same. Key generation is the same as with RSA-Small-e also, except that the roles of e and d are interchanged. Unlike the small public exponent case though, very small values of d cannot be used safely. As we shall see in the next section, the private exponent should be greater than about $N^{0.34}$ to be considered safe.

2.2. RSA-CRT

Based on the Chinese Remainder Theorem (CRT), Quisquater & Couvreur [19] proposed a fast decryption algorithm called CRT-decryption (described below). Combining this decryption method with a small public exponent leads to a significant decrease in encryption and decryption times as compared to the original RSA cryptosystem. We will call instances of RSA with a small public exponent using CRT-decryption **RSA-CRT**.

The key generation is the same as for RSA-Small-e, except that the private key is now the tuple (d_p, d_q, p, q), where $d_p \equiv d \pmod{p-1}$ and $d_q \equiv d \pmod{q-1}$. We call d_p and d_q CRT-exponents. When d_p and d_q are roughly the same size we say that they are balanced CRT-exponents.

Encryption is the same as the original definition. For a given ciphertext $C \in Z_N$, the decryption algorithm is as follows:

Step 1. Compute $C_p \equiv C^{d_p} \pmod p$.
Step 2. Compute $C_q \equiv C^{d_q} \pmod q$.
Step 3. Compute $M_0 \equiv (C_q - C_p)\cdot p^{-1} \pmod q$.
Step 4. Compute the plaintext $M = C_p + M_0 \cdot p$.

Note that the value of $p^{-1} \pmod q$ can be computed in advance (or can be added to the private key). Thus, the main cost for CRT decryption is in steps 1 and 2. Compared to the modular exponentiation in original RSA decryption, the bit-length of both the exponents and the modulus are half sized. Therefore, CRT decryption is approximately 4 times faster if the computations are done sequentially.

2.3. Rebalanced RSA-CRT

In order to further speed up decryption, Wiener suggested a variant of RSA-CRT which shifts some of the work done by decryption to encryption in [28]. Essentially, one chooses a private exponent d so that both d_p and d_q are small. We call this variant **Rebalanced RSA-CRT**.

Encryption and decryption are the same as with RSA-CRT. For key generation, to generate an instance Rebalanced RSA with an n-bit modulus, and n_d-bit CRT-exponents we do the following:

Step 1. Randomly select two distinct $n/2$-bit primes $p = 2p_1 + 1$ and $q = 2q_1 + 1$ such that $\gcd(p_1, q_1) = 1$.
Step 2. Compute $p_1^{-1} \pmod{q_1}$.
Step 3. Randomly select two distinct $n_d/2$-bit odd integers satisfying $\gcd(d_p, p-1) = 1$ and $\gcd(d_q, q-1) = 1$.
Step 4. Compute $d_0 \equiv (d_q - d_p)p_1^{-1} \pmod{q_1}$.
Step 5. Compute $d = d_p + d_0 \cdot p_1$.
Step 6. Compute $e \equiv d^{-1} \pmod{\phi(N)}$.

The public key is given by the pair (e, N) and the private key is given by the tuple (d_p, d_q, p, q). Note that e will be roughly the same order of magnitude as $\phi(N)$ with high probability. Thus, in order to reduce the decryption time even further than RSA-CRT, the encryption time is essentially maximized (the same as for original RSA encryption).

As with RSA-Small-d, in order to be secure the values of the CRT-exponents, d_p and d_q, cannot be chosen to be arbitrarily small. We discuss this more in the next section.

2.4. Generalized Rebalanced RSA-CRT

In this work we modify the key generation algorithm of Rebalanced RSA-CRT so that public exponents much

smaller than $\phi(N)$ can be used. In particular, we propose two schemes for key generation which will be given in section 4. Independently, a related scheme has been proposed by Galbraith, Heneghan & McKee [13]. The only significant difference between our schemes and their proposed variant is the key generation algorithm. Their scheme is slightly more general than the ones proposed here, since they do not restrict themselves to balanced primes and balanced CRT exponents, but their scheme only allows for public exponents smaller than $N^{0.5}$ when the primes are balanced. Another key generation algorithm has recently been proposed by Hinek [16], in which the RSA primes p and q satisfy $\gcd(p-1,q-1) = 2g$, where $g<N^{0.5}$ is a large prime. This scheme is more restrictive than the proposed schemes in this work and that of Galbraith, Heneghan & McKee. We will refer to all of these schemes as instances of Generalized Rebalanced RSA-CRT.

2.5. Notation

In the rest of this work we use the following notation: $N=pq$ is an n-bit RSA modulus with balanced primes (i.e., p and q are roughly the same size), e is an n_e-bit public exponent, $d\equiv e^{-1} \pmod{\phi(N)}$ is a valid private exponent for the public key (e, N), and k is the unique integer satisfying $ed=1+k\cdot\phi(N)$, called the key equation. When CRT-decryption is used, d_p and d_q are each n_d-bit CRT-exponents while k_p and k_q are each n_k-bit integers satisfying $ed_p=1+k_p\cdot(p-1)$ and $ed_q=1+k_q\cdot(q-1)$, respectively. We will call these equations the CRT key equations. Also, we often use $|x|$ to denote the bit-length of x.

3. Attacks on RSA and some Fast RSA Variants

In this section we give an overview of the strongest known algebraic attacks on RSA and some fast RSA variants. We do not consider non-algebraic attacks such as faults in software and side-channel attacks. Also, all of the attacks apply to instances of RSA (and variants) with balanced primes.

3.1. Original RSA

The original version of RSA used with an appropriate padding scheme, such as OAEP, is considered to be secure. In this case, with very high probability, both the public and private exponents are roughly the same order of magnitude as $\phi(N)$. While this essentially leads to maximized encryption and decryption times it also makes this version of RSA very resistant to all known attacks. By resistant, we mean that it is computationally infeasible to mount these attacks in the foreseeable future. When either the public or the private exponent is chosen in a special way however, such as with RSA-Small-e and RSA-Small-d, there exists attacks which are feasible to mount in some instances. These attacks are the partial key-exposure attacks and the small private exponent attacks [6, 9] (which can be considered as a partial key-exposure attack requiring no partial information). Essentially, given some number of the most significant bits (MSBs) or the least significant bits (LSBs) of the private exponent d, these attacks can recover the remaining bits of d or factor the modulus N; either resulting in a complete break of the system. These attacks include the partial key-exposure attacks of Boneh, Durfee & Frankel [1], Blömer & May [4] and Ernst, Jochemsz, May & de Weger [11] and the small private exponent attacks of Wiener [28] and Boneh & Durfee [3]. The effectiveness of all of the attacks, collectively, is illustrated nicely in the plots found in Section 1 of [11]. We replicate these plots below showing only the bounds for all attacks considered as one (for finer detail, we refer the reader to [11]). In each of the plots below, the y-coordinate represents the fraction of the private exponent that is needed to mount the attack. All values above each curve represent the region of parameters that are vulnerable to the attacks.

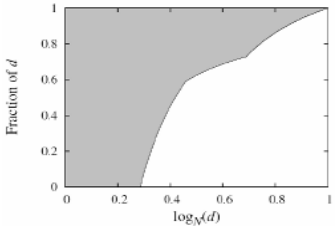


Figure 3.1: MSB attacks for small d.

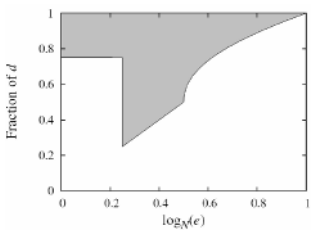


Figure 3.2: MSB attacks for small e.

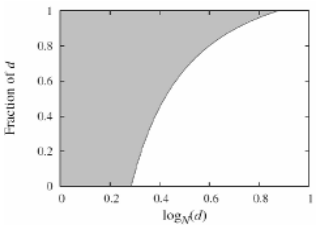


Figure 3.3: LSB attacks for small d.

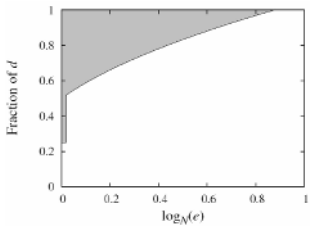


Figure 3.4: LSB attacks for small e.

As can be seen in the plots, when both the public and private exponents are the same order of magnitude as

$\phi(N)$, almost all of the LSBs or MSBs of d are needed to mount an attack. In the partial key-exposure attacks, the known bits of d can be obtained in different ways. For example, a side-channel attack may reveal the MSBs of d. Another way of obtaining these bits is by exhaustive search. An attacker can simply mount the attack with each possibility. Assuming that an adversary uses the latter method, we can then say something about the security of RSA with certain parameter choices (small e or d) in terms of how large the required search space is. For example, for a 1024-bit modulus, and a public exponent $e \approx N^{0.5}$, one would need to guess the 512 MSBs of d, which is certainly infeasible at this time. In fact, for small public exponents even the best partial key-exposure attacks require guessing 256 bits of d (LSBs) for a 1024-bit modulus. Thus, RSA-Small-e can be considered secure provided that no other method is available to obtain some of the bits of d (such as side-channel attacks). For small private exponents, on the other hand, if $d < N^{0.292}$, then the lattice-based attack of Boneh & Durfee [3] can, heuristically, recover d. It should be pointed out the result of Boneh & Durfee is asymptotic in the size of the modulus. That is, the bound on d in which the attack is guaranteed to work (under the assumption that the two equations found are algebraically independent) asymptotically approaches $N^{0.292}$ as N approaches infinity. And, this attack would require infinite computing power. In practice, the attack can be mounted on instances of RSA with slightly smaller private exponents in a reasonable amount of time. For example, Boneh & Durfee [3] successfully mount their attack on RSA with a 1000-bit modulus and $d \approx N^{0.280}$. The small private exponent attacks can be strengthened with a small exhaustive search as well. Let us consider Boneh & Durfee's first lattice-based attack in [3] in which $d < N^{0.284}$ can be recovered. In this attack, the goal is to find small solutions of the equation $x(N-y) \equiv 1 \pmod{e}$. Here, we look for the solution $(x_0, y_0) = (-k, p+q-1)$. Let us split $y_0 = p+q-1$ into its m MSBs and $(n/2-m)$ LSBs (recall that $p+q-1 = N-\phi(N) < 3N^{0.5}$ for RSA with balanced primes). That is, let $y_0 = y_1 2^m + y_2$, where $|y_2| < 2^m$. We can then use the same technique to solve the equation $x(N_0-z) \equiv 1 \pmod{e}$, where $N_0 = (N - y_1 2^m)$. Now we are looking for the solution $(x_0, z_0) = (-k, y_2)$. An attacker can exhaustively guess y_1 and mount the attack for each guess. Modifying the analysis in [3], we find that given the m MSBs of $(p+q-1)$ one can recover the private exponent for $e \approx N$ whenever

$$n_d < (7/6)n - (1/3)m - (1/3)((2m-7n)(2m-n))^{0.5}. \quad (3.1)$$

For a 1024-bit modulus (n=1024) and guessing 80 bits (m=80) the private exponent can be recovered if the bit-length of d is less than 348 (i.e., $d < N^{0.34}$). Therefore, we suggest that RSA-Small-d offers about the same security as factoring a 1024-bit RSA modulus provided that the bit-length of d is greater than about 350 bits (for a 1024-bit modulus). Using the analysis in [3] with geometrically progressive matrices, this bound could be increased by a few bits. Even so, the bound is fairly pessimistic since we are basing the security on the exhaustive search and ignoring the complexity of the lattice attack which is not insignificant.

3.2. RSA-CRT

In RSA-CRT, the public exponent is small ($e = 2^{16+1}$, say) while the private exponent ($d \equiv e^{-1} \bmod \phi(N)$) is with very high probability roughly the same order of magnitude as $\phi(N)$. Therefore, the small private exponent attacks of the previous section do not apply. The best known attack on RSA-CRT uses the Baby-Step Giant-Step method to compute one of the RSA primes [20]. This attack has time complexity $O(\min\{d_p^{0.5}, d_q^{0.5}\})$ for CRT-exponents d_p and d_q. In order to match the expected complexity of factoring a 1024-bit modulus, we enforce the complexity of this attack to be $O(2^{80})$ and so restrict d_p and d_q to be greater than about 160 bits. The CRT-exponents in RSA-CRT will have bit-length $n/2$ with very high probability (reducing an n-bit number modulo $n/2$-bit numbers will most likely result in numbers that roughly $n/2$-bits long). Therefore, this attack is not feasible for RSA-CRT. For example, RSA-CRT with a 1024-bit modulus will most likely have CRT-exponents that have bit-length 512.

3.3. Rebalanced RSA-CRT

In Rebalanced RSA-CRT, the CRT-exponents are chosen to be significantly smaller than $N^{0.5}$. In this variant, the Baby-Step Giant-Step attack [20] from the previous subsection might apply. Therefore, the CRT-exponents in Rebalanced RSA-CRT should have bit-length of at least 160 in order to match the security of factoring a 1024-bit RSA modulus. When the primes are unbalanced there exists a stronger attack by May [17].

3.4. Generalized Rebalanced RSA-CRT

Galbraith, Heneghan & McKee consider various attacks on Generalized Rebalanced RSA-CRT in [13]. We give an overview of their main attacks with respect to the parameters n_e, n_d, n and $n_k = |k_p| = |k_q|$. Each attack includes an exhaustive search on m bits. If k_p or k_q is known, they show that a lattice-based attack exists that can factor the modulus provided that $n_e \geq n/4-m$. Thus, to guard against this attack, one must ensure that $n_e \leq n/4-m$. When k_p and k_q are not known, there are two main attacks; a linearization attack and a lattice-based attack. The linearization attack recovers k_p and k_q whenever $n_k \leq (n_e+m)/3$. This attack is used in conjunction with the known k_p (or k_q) attack to ultimately factor the modulus. To guard against this attack, one must ensure that $n_k \geq (n_e+m)/3$, or at least ensure that $n_e \leq n/4-m$ so that the second part of the attack cannot be mounted. The last attack is a lattice-based attack that can be thwarted when $n_d + 4n_k \geq 2n_e + 4m$. The details of this last attack are omitted in [13].

In this work, we provide the details of the lattice-based attack mentioned in [13] and also show that their linearization attack can also be viewed as a lattice-based attack. The analysis in this work generates a slightly different condition for the last attack though (our analysis leads to a factor of m instead of $4m$).

4. Proposed Rebalanced RSA-CRT Schemes

In this section we present two key generation algorithms for Rebalanced RSA-CRT which generate public exponents much smaller than $\phi(N)$. The first key generation algorithm, Scheme A, is used to generate instances with public exponent $N^{0.5}<e<N$ and small CRT-exponents. The second algorithm, Scheme B, is used to generate instances with public exponent $e<N^{0.5}$ but with slightly larger CRT-exponents. Each key generation algorithm is based on the following fundamental theorem from number theory [18].

Theorem 4.1 Let a and b be relatively prime integers (i.e., gcd(a, b)=1). For every integer h there exists a unique and efficiently computable pair of integers (u_h, v_h) satisfying $au_h-bv_h=1$, where $(h-1)b<u_h<hb$ and $(h-1)a<v_h<ha$.

We now present the two key generation algorithms. The first algorithm can be used to generate instances of Rebalanced RSA-CRT with public exponents larger than $N^{0.5}$ ($n_e>n/2$), while the second algorithm generates instances with public exponents smaller than $N^{0.5}$ ($n_e<n/2$). Parameter suggestions and security considerations will be made in Section 5.

4.1. The Proposed Scheme A

The first key generation algorithm, scheme A, takes (n, n_e, n_d) as input, with $n_e>n/2$, and outputs a valid public key (e, N) and corresponding private key (d_p, d_q, p, q), where $|N|=n$, $|e|=n_e$ and $|d_p|=|d_q|=n_d$. The algorithm is as follows:

Input: (n, n_e, n_d), where $n_e>n/2$.

Step 1. $e \leftarrow$ random (n_e)-bit odd integer.
Step 2. $k_{p1} \leftarrow$ random (n_d)-bit integer such that gcd(k_{p1}, e)=1.
Step 3. Using Theorem 4.1 (with h=2), compute (d_p, P) satisfying $ed_p=k_{p1}P+1$, where $k_{p1}<d_p<2k_{p1}$ and $e<P<2e$.
Step 4. Factor P as $P=k_{p2}(p-1)$ where k_{p2} is an ($n_e-n/2$)-bit integer and p is prime. If this is infeasible, then go to Step 2.
Step 5. $k_{q1} \leftarrow$ random (n_d)-bit integer such that gcd(k_{q1}, e)=1.
Step 6. Using Theorem 4.1 (with h=2), compute (d_q, Q) satisfying $ed_q=k_{q1}Q+1$, where $k_{q1}<d_q<2k_{q1}$ and $e<Q<2e$.
Step 7. Factor Q as $Q=k_{q2}(q-1)$ where k_{q2} is an ($n_e-n/2$)-bit integer and q is prime. If this is infeasible, then go to Step 5.

Output: the public key (e, N) and the private key (d_p, d_q, p, q).

Now we show that the output is indeed a valid public key and corresponding private key. Let $k_p=k_{p1}k_{p2}$ and $k_q=k_{q1}k_{q2}$. From steps 2~4 and 5~7, we know that $ed_p=k_p\cdot(p-1)+1$ and $ed_q=k_q\cdot(q-1)+1$, respectively. This implies gcd(e, $\phi(N)$)=1. Furthermore, multiplying these two equations together yields $(ed_p-1)(ed_q-1)=k_p(p-1)\cdot k_q(q-1)$, which can be rewritten as $e(-ed_pd_q+d_p+d_q)=-k_pk_q(p-1)(q-1)+1$. Letting $d_0=-ed_pd_q+d_p+d_q$ and $k_0=-k_pk_q$, we see that e and d_0 satisfy the equation $ed_0=k_0\cdot\phi(N)+1$. Thus, e and d_0 are a valid public/private exponent pair for the modulus $N=pq$. It is not a problem that both d_0 and k_0 are negative. In fact, (e, $d_0+\alpha\cdot\phi(N)$) is also a valid public/private exponent pair for any integer α. And, for one particular value of α, say α_0, we have $d=d_0+\alpha_0\cdot\phi(N)$ where $0<d<\phi(N)$. Thus, the key generation algorithm can be seen to, essentially, output a valid RSA public key (e, N) with corresponding private key (d, N). Now, $d=-ed_pd_q+d_p+d_q+\alpha_0\cdot\phi(N)=-(1+k_p\cdot(p-1))d_q+d_p+d_q+\alpha_0\cdot\phi(N)\equiv -d_q+d_p+d_q\equiv d_p \pmod{p-1}$ and, similarly, $d\equiv d_q \pmod{q-1}$, so (d_p, d_q, p, q) is also a valid private key corresponding to the public key (e, N).

Notice that the algorithm basically consists of two loops: steps 2~4 and steps 5~7. Therefore, the complexity is linear in the expected number of iterations for each of the loops. It has been experimentally observed that the number of iterations for each loop is about 2^{15}. Also, in each loop we try to factor an (n_e)-bit integer (P or Q) into an ($n_e-n/2$)-bit number and an ($n/2$)-bit number. In order to factor this efficiently, using ECM for example, ($n_e-n/2$) must not be too large. Therefore, we must restrict the choice of n_e (public exponent size) to be close enough to $n/2$ so this factoring problem is feasible with the resources available.

4.2. The proposed Scheme B

The second key generation algorithm, scheme B, takes (n, n_e, n_d) as input, with $n_e<n/2$, and outputs a valid public key (e, N) and corresponding private key (d_p, d_q, p, q), where $|N|=n$, $|e|=n_e$ and $|d_p|=|d_q|=n_d$. The algorithm is as follows:

Input: (n, n_e, n_d), where $n_e<n/2$

Step 1. $e \leftarrow$ random (n_e)-bit odd integer.
Step 2. $d_{p1} \leftarrow$ random ($n/2-n_e$)-bit odd integer, $E_p \leftarrow ed_{p1}$.
Step 3. $k_p \leftarrow$ random ($n_e+n_d-n/2$)-bit integer satisfying gcd(k_p, E_p)=1.
Step 4. Using Theorem 4.1 (with h=2), compute (d_{p2}, p_1) satisfying $E_pd_{p2}=k_p\,p_1+1$, where $k_p<d_{p2}<2k_p$ and $E_p<p_1<2E_p$, and let $d_p \leftarrow d_{p1}d_{p2}$.

Step 5. $p \leftarrow p_1+1$. If p is not prime, then go to Step 3.
Step 6. $d_{q1} \leftarrow$ random $(n/2-n_e)$-bit odd integer, $E_q \leftarrow ed_{q1}$.
Step 7. $k_q \leftarrow$ random $(n_e+n_d-n/2)$-bit integer satisfying $\gcd(k_q, E_q)=1$.
Step 8. Using Theorem 4.1 (with h=2), compute (d_{q2}, q_1) satisfying $E_q d_{q2}=k_q q_1+1$, where $k_q<d_{q2}<2k_q$ and $E_q<q_1<2E_q$, $d_q \leftarrow d_{q1}d_{q2}$.
Step 9. $q \leftarrow q_1+1$. If q is not prime, then go to Step 7.

Output: the public key (e, N) and the private key (d_p, d_q, p, q).

That (e, N) and (d_p, d_q, p, q) is a valid public/private key pair follows from, essentially, the same reasoning as in Scheme A and so we omit the details here.

The key generation algorithm for this scheme is much more efficient than the one in Scheme A. The runtime of the algorithm is dominated by two loops, like Scheme A, but each iteration requires much less computational effort as there is no factoring required. And, the experimentally observed expected number of iterations for each loop is much less also. It was observed that each loop requires about 2^{10} iterations for this algorithm.

4.3. Sparse Public Exponents

In both schemes the public exponent e is chosen before the RSA primes are generated. In order to speed up encryption further, we can modify step 1 in each algorithm to choose an (n_e)-bit odd integer that has a sparse binary representation. In particular, we can use $e=2^{n_e-1}+1$, which minimizes the number of multiplications required for encryption. In the original Rebalanced RSA-CRT, since the public exponent is computed as the inverse of a number modulo $\phi(N)$ it will be, with high probability, roughly the same order of magnitude as $\phi(N)$ and its binary representation will on average have an equal number of ones and zeros. Since $|\phi(N)|=|N|=n$, the average number of multiplications (including squarings) needed for encryption will be $1.5\times n$. Using a sparse public exponent generated with either scheme reduces the number of multiplications to essentially n_e. When compared to the original Rebalanced RSA-CRT, using a sparse public exponent with Scheme A results in a speed-up for encryption by a factor of at most 3 (and at least 1.5), while using a sparse public exponent with Scheme B results in a speed-up for encryption by a factor of at least 3. Therefore, we suggest that sparse public exponents always be used in these schemes.

Scheme A is designed to generate instances of Rebalanced RSA-CRT with public exponents larger than $N^{0.5}$. The key generation algorithm for Scheme A is very time consuming though. In Appendix B, we present an alternate scheme, denoted Scheme A^+, that also generates instances of Rebalanced RSA-CRT with public exponents larger than $N^{0.5}$ that has a more efficient key generation algorithm. Unfortunately, Scheme A^+ does not allow the user to generate sparse public exponents.

5. Security Analysis

In this section we focus on the security of Generalized Rebalanced RSA-CRT. In particular, we investigate which combinations of n_e and n_d are secure and which are insecure. We begin by first considering the security of Rebalanced RSA-CRT when viewed as normal RSA and then consider attacks that exploit the CRT key equations. The results of the latter attacks are essentially the same results in [13]. Here, we provide the details of their lattice-based attack and also show that their linearization attack can be viewed as a lattice-based attack as well. In addition, we also present two new attacks that slightly reduce the secure region for the parameter choices for the public and private exponents (n_e and n_d).

5.1. Generalized Rebalanced RSA-CRT as Normal RSA

Multiplying together the two CRT key equations, written as $ed_p-1=k_p(p-1)$ and $ed_q-1=k_q(q-1)$, yields (after some rearrangement)

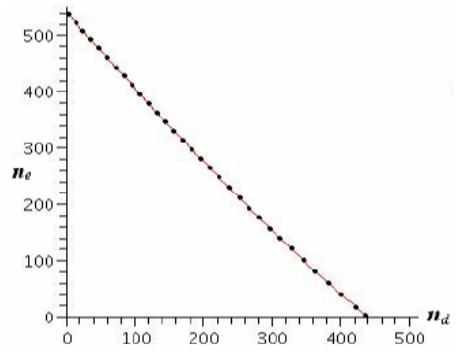
$$e(d_p+d_q-ed_pd_q)=1-k_pk_q\phi(N) \tag{5.1}$$

which defines a valid RSA public/private exponent pair $(e, d_0=d_p+d_q-ed_pd_q)$ with constant $k_0=-k_pk_q$. We therefore must consider the security of this public/private exponent pair with respect to all the known attacks on RSA (from section 3). We begin with the partial key-exposure attacks. From the CRT key equations we have $n_e+n_d \geq n/2$ and so $|d_0| \approx n_e+2n_d \geq n/2$. Therefore, each of the small private exponent partial key-exposure attacks require knowing at least $n/4$ bits of d_0 (since at least 1/2 the bits of d_0 must be known). For the small public exponent attacks when $n_e \geq n/2$, again, at least $n/4$ bits of d_0 must be known. Some stronger attacks exist when $n_e \leq n/2$ which only require 1/4 of the bits of d_0 in the best case (from Boneh, Durfee & Frankel [1]). In the known LSBs attack, n_e is very small and so $n_d \approx n/2$. Therefore, $|d_0| \approx n$ and so $n/4$ bits of d_0 are required. For known MSBs, the attack is strongest when $n_e=n/4$. In this case, $n_d \geq n/4$ and so we have $|d_0| \approx 3n/4$. Therefore, at least $3n/16$ bits of d_0 are required. Considering all the partial key-exposure attacks, we then see that at least $3n/16$ bits of d_0 are required (for a 1024-bit modulus this is 192 bits). Guessing these bits with an exhaustive search is as expensive as factoring a 1024-bit RSA modulus whenever $n>428$. So, if we make the assumption that the implementation of the cryptosystem is resistant to side channel attacks we then suggest that the partial key-exposure attacks are not a threat to the security of the proposed schemes. It should be pointed out that in some instances $|d_0|>n$. For these instances, the partial

key-exposure attacks apply to d_0 modulo $\phi(N)$ instead of d_0. This is not a threat, however, since in these instances d_0 modulo $\phi(N)$ will be roughly the same order of magnitude as $\phi(N)$ with very high probability.

The small private exponent attacks on RSA present a more serious danger. We consider Boneh & Durfee's lattice-based attack [3, 20]. Let $\Lambda=N-\phi(N)=\Lambda_m+\Lambda_0$ and $M=N-\Lambda_m$, where Λ_m represents the m MSBs of Λ_m and Λ_0 represents the remaining bits. With this we can write the key equation as $ed_0=1+k_0(M-\Lambda_0)$. Assuming that M is known (i.e., Λ_m is known), the lattice-based small private exponent attacks involve trying to find all integer pairs (x_0, y_0) such that $|x_0|<X$, $|y_0|<Y$ and $f(x_0, y_0)=x_0(M-y_0)\equiv 1$ modulo e, where the bounds X and Y are defined so that $X=|k_0|$ and $Y=|\Lambda_0|$. One such pair will be $(x_0, y_0)=(-k_0, \Lambda_0)$. Once we know Λ_0 we also know $\phi(N)$ which then allows us to invert e modulo $\phi(N)$; which breaks the system. Since $\log_2(e)=n_e$, the bounds become $X=e^{(2n_e + 2n_d - n)/ n_e}$ and $Y=e^{(n/2 - m)/n_e}$. Using equation (3.1), we find that a sufficient condition for the attack to succeed, in the limit of large n, is given by

$$n_d < 7n/12 - m/6 - n_e/2 - (1/6)(4m^2 - 4mn - 12mn_e + 6n_e n)^{0.5}. \quad (5.2)$$



In Figure 5.1, we see the effectiveness of the attack for a 1024-bit modulus when we know the 80 high bits of Λ. Any combination of n_e and n_d below the curve in the figure are vulnerable to the attack. Assuming that the high bits of Λ are obtained by an exhaustive search and that the attack is mounted once for each guess, any combination of n_e and n_d above the curve requires an expected amount of work roughly the same as factoring a 1024-bit RSA modulus (i.e., $O(2^{80})$). Therefore, instances of Rebalanced CRT-RSA in which inequality (5.2) is violated can be considered secure.

Figure 5.1: Effectiveness of Boneh & Durfee's small private exponent attack on Rebalanced RSA-CRT. The region under the curve represents instances in which the small private exponent lattice-based attacks can succeed for a 1024-bit modulus when 80 bits of Λ are known.

5.2 Generalized Rebalanced CRT-RSA

First we consider the best known attack on RSA-CRT. This is the Baby-Step Giant-Step attack of Qiao & Lam [20]. The attack has complexity $O(\min\{d_p^{0.5}, d_q^{0.5}\})$ for CRT-exponents d_p and d_q. To match the expected (heuristic) complexity of factoring a 1024-bit modulus, we require the complexity to be $O(2^{80})$ and so require $n_d \geq 160$. This is the same restriction that is applied to RSA-CRT. In general we require $n_d \geq 2m$ so that the Baby-Step Giant-Step attack has complexity comparable to an exhaustive search on m bits.

Next we consider lattice-based attacks. Multiplying together the two CRT key equations, written this time as $ed_p-1+k_p=k_p p$ and $ed_q-1+k_q=k_q q$, yields (after some rearrangement)

$$e^2 d_p d_q + e(d_p(k_q-1)+d_q(k_p-1)-k_p k_q(N-1)-k_p-k_q+1=0 \quad (5.3)$$

Let k_m represent the m most significant bits of $-(k_p+k_q)+1$ and let k_0 represent the remaining bits (i.e., $k_m+k_0=-(k_p+k_q)+1$). Assuming that we know k_m, by guessing for example, we can rewrite equation (5.3) as

$$e^2 d_p d_q + e(d_p(k_q-1)+d_q(k_p-1))-k_p k_q(N-1)+k_m+k_0=0 \quad (5.4)$$

where d_p, d_q, k_q, k_p and k_0 are unknown. Using lattice basis reduction techniques based on Coppersmith's method we try to solve this equation when the unknowns are relatively small. Since k_m is generally not known, we will perform an exhaustive search for k_m mounting the lattice-based attack for each guess. There are several ways in which we can mount a lattice based attack starting with equation (5.4). Indeed, equation (5.4) can be viewed as a linear equation in 4 variables over the integers, as a linear equation in 3 variables modulo e^2, as a linear equation in 2 variables modulo e, and finally as a linear equation in 3 variables modulo N. As we shall see, each leads to different instances of n_e and n_d which are (in)secure. In each case we are looking for small solutions of linear equations. Equation (5.3) can also be viewed in many ways as nonlinear equations also, but we do not concern ourselves with these as they lead to weaker attacks compared to the linear equations. The main results that we will use, given below, concern finding small solutions to linear modular equations.

Theorem 5.1 [Trivariate Linear Modular Equations]: Let $f(x, y, z)$ be a linear polynomial with integer coefficients. For every $\varepsilon>0$, there exists a positive M_0 such that for every integer $M>M_0$ that is relatively prime to at least one non-constant coefficient of f we can find three linearly independent polynomials such that each root (x_0, y_0, z_0) of $f(x, y, z)$ modulo M is also a root of each the three polynomials modulo M, and if $|x_0|<X$, $|y_0|<Y$, $|z_0|<Z$ and $XYZ<M^{1-\varepsilon}$, for some bounds X, Y and Z, then (x_0, y_0, z_0) is also a root of each of the three polynomials over the integers. If these three polynomials are also algebraically independent then we can compute (x_0, y_0, z_0).

Corollary 5.2 [Bivariate Linear Modular Equations]: Let $f(x, y)$ be a linear polynomial with integer coefficients. For every $\varepsilon>0$, there exists a positive M_0 such that for every integer $M>M_0$ that is relatively prime to at least one non-constant coefficient of f we can find two linearly independent polynomials such that each root (x_0, y_0) of $f(x, y)$

modulo M is also a root of each the two polynomials modulo M, and if $|x_0|<X$, $|y_0|<Y$ and $XY<M^{1-\varepsilon}$, for some bounds X and Y, then (x_0, y_0) is also a root of each of the two polynomials over the integers. If these two polynomials are also algebraically independent then we can compute (x_0, y_0).

It is currently an open question to prove that the polynomials obtained are algebraically independent or not, but in practice if the solution is not too close to the bounds it is often the case that the polynomials are algebraically independent. The proof of Theorem 5.1 and corollary 5.2 will be given in section 5.4. We now simply use these results to show which combinations of n_e and n_d are (in)secure.

The strongest attack is obtained by considering equation (5.3) modulo e^2. This is the lattice-based attack in Section 5.2 of [13]. Here we look for small solutions of $f(x, y, z)=ex-(N-1)y+z+k_m \pmod{e^2}$. In particular, we are looking for $(x_0, y_0, z_0)=(d_p(k_q-1)+d_q(k_p-1), -k_pk_q, k_0)$. Notice that y_0 and z_0 yield k_p and k_q. The remaining values (d_p and d_q) are recovered by substituting x_0, y_0 and z_0 into equation (5.3) to obtain a new equation with unknown variable d_pd_q. Using k_p, k_q, the equation for x_0 and this new equation allows us to recover d_p and d_q (since we now have two equations in two variables). Using the bounds X, Y and Z given by $\log_2(X)=2n_d+n_e-n/2$, $\log_2(Y)=2n_d+2n_e-n$, and $\log_2(Z)=n/2-m$ with Theorem 5.1, we obtain the following enabling equation to find three linearly independent polynomials with the root (x_0, y_0, z_0) over the integers:

$$5n_d+2n_e<2n+m \tag{5.5}$$

Thus, if we choose n_e and n_d so that inequality (5.5) is satisfied, we can expect to do at least $O(2^m)$ work to recover d_p and d_q (which is a full break of the system). With m=80, we then expect to do at least as much work as factoring a 1024-bit RSA modulus. Also, if we substitute $n_k=n_d+n_e-n/2$, which holds when the primes are balanced, into the lattice-based attack in Section 5.2 of [13] we obtain, essentially, the same result.

If we instead consider equation (5.3) modulo e, we then look for small solutions to $g(x, y)=(N-1)x+y+k_m \pmod{e}$. In particular, we are looking for the root $(x_0, y_0)=(-k_pk_q, k_0)$. In this case, finding x_0 and y_0 only yields k_p and k_q though. Using the bounds X and Y defined by $\log_2(X)=2n_d+2n_e-n$ and $\log_2(Y)=n/2-m_1$ with Corollary 5.2, we obtain the following enabling equation to find two linearly independent polynomials with the root (x_0, y_0) over the integers:

$$3n_d+2n_e<(3/2)n+m_1 \tag{5.6}$$

As above, if we choose n_e and n_d so that inequality (5.6) is satisfied, we can expect to do at least $O(2^{m_1})$ work to recover k_p and k_q. Once k_p and k_q are known, we try to factor the modulus using the known k_p (or k_q) attack from Galbraith, Heneghan & McKee [13]. This factoring attack works can be mounted when $n_e>n/4-m_2$, where another exhaustive search must be carried out on m_2 bits. To impose $O(2^m)$ work for the exhaustive searches we simply let $m=m_1+m_2$. Notice that if we substitute $n_k=n_d+n_e-n/2$, which holds when the primes are balanced, into the linearization attack in Section 5.1 of [13] we obtain the same result. Thus, we have shown that Galbraith, Heneghan & McKee's linearization attack is equivalent to a lattice-based attack.

Lastly, we consider reducing equation (5.2) modulo N. Here we look for small solutions of $h(x, y, z)=e^2x+ey+z+k_m^* \pmod{N}$, where k_m^* represents the m most significant bits of $k_pk_q-k_p-k_q+1$ (and k_0^* represents the remaining bits). In particular, we look for the root $(x_0, y_0, z_0)=(d_pd_q, d_p(k_q-1)+d_q(k_p-1), k_0^*)$. All of d_p, d_q, k_q and k_p can be recovered with knowledge of (x_0, y_0, z_0). Each of x_0, y_0 and z_0 yields an equation in (d_p, d_q, k_q, k_p) and substituting (x_0, y_0, z_0) into equation (5.3) will yield an additional equation with unknown k_pk_q. Thus, we have four equations and four unknowns which we can solve. For very small n_e values, this attack is slightly superior to the mod e^2 attack and the mod e attack. In this region, the mod e attack can only recover k_p and k_q though. Using the bounds $X=2^{2n_d}$, $Y=2^{2n_d+n_e-n/2}$, and $Z=2^{2n_d+n_e-n-m}$ with Theorem 5.1 we obtain the following enabling equation to find three linearly independent polynomials with the root (x_0, y_0, z_0) over the integers:

$$6n_d+3n_e<(5/2)n+m \tag{5.7}$$

As with the previous cases, if we choose n_e and n_d so that inequality (5.7) is satisfied, we can expect to do at least $O(2^m)$ work to recover k_p and k_q

5.3 Secure Parameter Choices

Here we summarize the attacks presented above, by giving conditions on n_e and n_d so that each attack requires at least $O(2^m)$ work. Using m=80 results in work that is comparable to factoring a 1024-bit RSA modulus. To be considered secure, it is recommended that each inequality given below be satisfied:

$5n_d+2n_e>2n+m$	(mod e^2)
$3n_d+2n_e>(3/2)n+m_1$ or $n_e<n/4-m_2$ $\quad (m=m_1+m_2)$	(mod e / known k_p)
$6n_d+3n_e>(5/2)n+m$	(mod N)
$6n_d+3n_e>(7/3)n-m-(1/2)(4m^2-4mn-12mn_e+6n_en)^{0.5}$	(small private exponent)
$n_d>2m$	(Baby-Step Giant-Step)

In Figure 5.2 we illustrate each of the above inequalities for a 1024-bit modulus with m=80. Any parameter choice for n_e and n_d in the shaded region violates one or more of the inequalities and should be considered insecure. The region between the dotted line and the shaded region represents parameter choices in which the mod e attack

recovers k_p and k_q but there is no known method to recover d_p and d_q (or factor N) with this knowledge. And finally, the remaining region above and to the right of all the lines represents parameter choices that can be considered secure. In order to simplify the visual representation we have been overly pessimistic and let $m_1=m_2=m$ instead of considering all combinations of m_1 and m_2 such that $m_1+m_2=m$. Notice that the small private exponent attack (Section 5.1) and the mod N lattice-based attack slightly extend the shaded (insecure) region of the previously known attacks (mod e and mod e^2). This region can be seen at the bottom-right of the shaded region when $n_d \approx 400$ and n_e is small (<200). Also notice that the small private exponent attack is implied by the other attacks (i.e., the region corresponding to this attack lies completely within the shaded region of the other attacks).

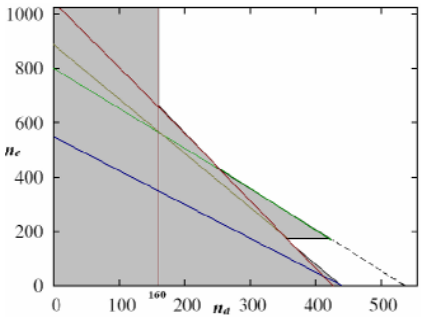


Figure 5.2: Lattice attacks on Rebalanced RSA-CRT.

5.4 Small Solutions of Linear Equations

In this subsection, we give a proof for Theorem 5.1. First we recall two important results. The first gives a bound on the m-th smallest vector of an LLL-reduced lattice basis [15].

Theorem 5.3 (LLL) Let $b_1, \ldots, b_n$ be an LLL-reduced basis of a full di mensional integral lattice $\boldsymbol{L} \in Z^n$. For m=1 or $1<m\le n$ and $|b_m| \ge 2^{(n-2)/2}$, then we have the following property: $|b_m| \le 2^{(n+m-2)/4} \det(\boldsymbol{L})^{1/(n-m+1)}$.

The next result gives a sufficient condition for roots of a polynomial modulo some integer to also be roots of the polynomial over the integers. The result is usually attributed to Howgrave-Graham [12]. We give a slightly generalized version of it.

Theorem 5.4 (Howgrave-Graham) Let $h(x_1,..., x_n) \in Z[x_1,..., x_n]$ be the sum of at most w monomials and let m be a positive integer. Suppose that there exists $y_1,..., y_n \in Z$ such that $h(y_1,..., y_n) \equiv 0 \pmod{M^m}$ and $\|h(x_1X_1,...,x_nX_n)\|_2 < w^{0.5}M^m$, where $|y_1|<X_1,..., |y_n|<X_n$ for some positive bounds $X_1,..., X_n$. Then $h(y_1,..., y_n)=0$ over the integers.
We now give a proof of Theorem 5.1.

***Proof of Theorem 5.1*:** Let M be a positive integer with unknown factorization and consider the linear integer polynomial $f(x, y, z)=Ax+By+Cz+D \in Z[x, y, z]$, such that at least one non-constant coefficient is relatively prime to M. We will assume, without loss of generality, that $\gcd(C, M)=1$. Multiplying $f(x, y, z)$ by C^{-1} modulo M we obtain the new polynomial $f_M(x, y, z)=ax+by+z+c$, which has the same roots as $f(x, y, z)$ modulo M. We will look for all small roots of $f_M(x, y, z)$ modulo M. That is, given bounds X, Y, Z, we wish to find all $(x_0, y_0, z_0) \in Z^3$ such that $|x_0|<X$, $|y_0|<Y$, $|z_0|<Z$ and $f_M(x_0, y_0, z_0) \equiv 0 \pmod{M}$. In the following, let (x_0, y_0, z_0) be any such root. We will try to find the maximal bounds that lattice basis reduction techniques (based on Coppersmith's techniques) can allow.

	i, j, k	-	x	y	z	x^2	xy	y^2	xz	yz	z^2	...
D=0	0,0,0	M^m										
	1,0,0		M^mX									
D=2	0,1,0			M^mY								
	0,0,1	*	*	*	$M^{m-1}Z$							
	2,0,0					M^mX^2						
	1,1,0						M^mXY					
	0,2,0							M^mY^2				
D=3	1,0,1		*			*	*		$M^{m-1}XZ$			
	0,1,1			*			*	*		$M^{m-1}YZ$		
	0,0,2	*	*	*	*	*	*	*	*	*	$M^{m-2}Z^2$	
...	…											...

Table 1: The matrix spanned by $g_{i,j,k}$ for i=0,…, k and $j=k-i$ and k=0,…, m. In the basis, '*' denotes some non-zero term while blank entries are zero terms.

For some positive integer m, to be determined later, we consider the set of polynomials $g_{i,j,k}(x, y, z)=$

$x^i y^j M^{m-k} f_M^k(x, y, z)$, for non-negative integer values of i, j, k. Notice that for any values of i, $j \geq 0$ and $0 \leq k \leq m$, we have $g_{i,j,k}(x_0, y_0, z_0) \equiv 0 \pmod{M^m}$. We construct a lattice basis for a lattice $\boldsymbol{L}$ using the coefficients of $g_{i,j,k}(xX, yY, zZ)$ for certain values of i, j, k. In particular, for each $0 \leq d \leq m$ we use all values $0 \leq k \leq d$ along with each combination of i and j such that $i+j \leq d-k$. This gives $w=m(m+1)(m+2)/6$ linearly independent vectors which form the lattice basis for $\boldsymbol{L}$. The structure of the basis matrix is shown below in Table 1 (lower right-hand side of the table).

Each row in the basis matrix is the coefficient vector of one of the $g_{i,j,k}(xX, yY, zZ)$. Notice that the ordering of the columns ensures that the basis matrix is triangular and that the lattice is full dimensional (with dimension w). Since the matrix is triangular, we can easily compute the determinant/volume of the lattice (given by the absolute of the determinant of the basis matrix) yielding $\det(\boldsymbol{L})=(XYZM^3)^{m(m+1)(m+2)(m+3)/24}=(XYZM^3)^{w(m+3)/4}$. Applying the LLL-algorithm to the above lattice we can find three linearly independent vectors in $\boldsymbol{L}$ whose sizes are bound by $2^{(w-1)/4}(\det(\boldsymbol{L})^{1/(w-2)})$ (from Theorem 5.3), where w is the lattice dimension defined above. Each of these vectors, being elements of $\boldsymbol{L}$, correspond to a polynomial of the form $h(xX, yY, zZ)$ such that $h(x_0, y_0, z_0) \equiv 0 \pmod{M^m}$. Let these polynomials be denoted by h_1, h_2, and h_3. Using Howgrave-Graham's condition for integer roots (Theorem 5.4), we see that a sufficient condition for (x_0, y_0, z_0) to be a root of these polynomials over the integers rather modulo M^m is given by $|h_i| < 2^{(w-1)/4}\det(\boldsymbol{L})^{1/(w-2)} < M^m/w^{1/2}$, which simplifies to $\det(\boldsymbol{L}) < M^{m(w-2)}/(2^{(w-1)(w-2)/4} w^{(w-2)/2})$. Substituting the determinant into this bound, and rearranging, gives

$$XYZ < M^{4m(w-2)/(w(m+3))-3} / (2^{(w-1)(w-2)/(w(m+3))} w^{2(w-2)/(w(m+3))}) = c_{m,w} M^{1-\varepsilon} \tag{5.7}$$

where $c=1/(2^{(w-1)(w-2)/(w(m+3))} w^{2(w-2)/(w(m+3))})$ and $\varepsilon=12(m^2+3m+6)/(m^3+6m^2+11m+6)$. For large enough m and w, we obtain the asymptotic result $XYZ<M$ as $M \to \infty$. Of course, whenever equation (5.7) is satisfied we can only guarantee that three linearly independent polynomials having the integer root (x_0, y_0, z_0) can be found. If the polynomials are further algebraically independent, we can use resultants to remove variables. For example, using h_1 and h_2 we compute $h_{1,2}(x, y)=\mathrm{Res}_z(h_1, h_2)$ and using h_1 and h_3 we compute $h_{1,3}(x, y)=\mathrm{Res}_z(h_1, h_3)$. These new polynomials will each have the integer root (x_0, y_0). We then compute the polynomial $h_{1,2,3}(x)=\mathrm{Res}_y(h_{1,2}, h_{1,3})$ which has the root x_0. Using standard techniques, we can solve for x_0. Solving the univariate polynomial $h_{1,2}(x_0, y)$ we can find y_0 and finally we obtain z_0 from solving $h_1(x_0, y_0, z)$. This completes the proof.

A simple proof of Corollary 5.2 follows directly from Theorem 5.1. For a given linear bivariate polynomial $f(x,y)$ with desired solution (x_0, y_0) we can simply consider the linear trivariate polynomial $g(x, y, z)=f(x, y)+z-1$, with desired root $(x_0, y_0, 1)$ and apply Theorem 5.1. It is straightforward and very similar to the proof of Theorem 5.1 to prove Corollary 5.2 using only two variables so we omit the details of this proof to save space.

In section 5.2 we also mention that an attack can be mounted by considering equation (5.3) as an integer linear equation in four variables. This attack yields the same result as considering equation (5.3) reduced modulo e^2, so we omit the details.

5.5 This Work and Tunable Balancing of RSA

In this work we have provided the details of the mod e^2 lattice-based attack in Section 5.2 of [13], shown that their linearization attack is equivalent to a mod e lattice-based attack and provided two additional attacks that slightly reduce the region of secure parameter choices (for n_e and n_d).

6. Implementations and Comparisons

To demonstrate the feasibility of the proposed schemes we have implemented the key generation algorithms and measured the average running time for each. A personal computer (PC) with 2.5 GHz CPU and 512 MB DRAM was used. The algorithms were implemented using NTL [22] with GMP [14] using Cygwin tools on the Windows operating system. For Scheme A we ran the key generation algorithm 100 times obtaining an average key generation time of 1682 seconds. The average number of iterations for each loop (steps 2~4 & 5~7) was 75312 ($\approx 2^{16}$). For the factoring operations in steps 4 & 7 we used the ECM. Due to the large key generation times this Scheme can only be utilized when key generation can be done offline. In our implementation the complete prime factorization of P (and Q) was not computed. Instead, we only try to find small factors until the desired factorization ($P=k_{p2}(p-1)$) is found. For Scheme B, the average key generation time was only 93 ms while the average number of iterations for each loop (steps 3~5 & 7~9) was 842. Thus, the key generation for this scheme is very efficient. Of course, decryption times for Scheme B are longer than decryption time for Scheme A. Thus, there is a trade-off between Scheme A and Scheme B.

In Table 2, we compare an instance of Scheme A and Scheme B with the other RSA variants in terms of the complexity of encryption and decryption for a 1024-bit modulus with security parameter m=80. Here we assume that a random k-bit exponent will require $1.5 \times k$ multiplications. We also assume that decryption is always carried out using the Chinese Remainder Theorem. Thus, RSA-small-e is the same as the original RSA-CRT. The complexity for each operation (encryption and decryption) is the expected number of bit operations. To compute $Z^a \pmod{b}$ for a random exponent a, we expect $1.5 \times |a| \times |b|^2$ binary operations [10]. When the exponent is of the special form of 2^k+1, then the number of binary operations will be reduced to $(k+1) \times |b|^2$ bit operations. In each of the instances for Scheme A and Scheme B, we use public exponents of the form 2^k+1. For Scheme A we use the instance $(n_e, n_d)=(592, 190)$ and for Scheme B we use the instance $(n_e, n_d) \approx (170, 358)$. Both instances satisfy each of the security conditions listed in section 5.3. Compared to the original Rebalanced RSA-CRT our instance from

Scheme A encrypts about 2.6 times faster while decryption is about 1.2 times slower. For Scheme B, compared to the original Rebalanced RSA-CRT, encryption is about 9 times faster while decryption is about 2.2 times slower.

	RSA-small-e	RSA-Short-d	RSA-CRT	Rebalanced RSA-CRT	Scheme A	Scheme B
Public Exponent	$e=2^{16}+1$	n *bits*	$e=2^{16}+1$	n *bits*	$e=2^{591}+1$	$e=2^{169}+1$
Number of Multiplications	$0.0166\times n$	$1.5\times n$	$0.0166\times n$	$1.5\times n$	$0.58\times n$	$0.16\times n$
Complexity	$17\times n^2$	$1536\times n^2$	$17\times n^2$	$1536\times n^2$	$592\times n^2$	$170\times n^2$
CRT-Exponent	--	--	$0.5\times n$ *bits*	>160 *bits*	190 *bits*	358 *bits*
Private Exponent	n *bits*	> 350 *bits*	--	--	--	--
Modulus	n *bits*	n *bits*	$0.5\times n$ *bits*	$0.5\times n$ *bits*	$0.5\times n$ *bits*	$0.5\times n$ *bits*
Number of Multiplications	$1.5\times n$	$> 1.5\times 350$	$1.5\times n$	$> 0.469\times n$	$0.55\times n$	$1.04\times n$
Complexity	$1536\times n^2$	$> 525\times n^2$	$384\times n^2$	$> 120\times n^2$	$142\times n^2$	$268\times n^2$

Table 2: Comparisons of the different RSA variants in terms encryption and decryption times for an n=1024 bit modulus with m=80.

7. Conclusions

In this work we have presented two key generation algorithms for Generalized Rebalanced RSA-CRT, which allow the size of the public and CRT exponents to be chosen by the user. The first scheme, Scheme A, generates public exponents with $n/2<n_e<n$ and CRT exponents that are fairly close that of the original Rebalanced RSA-CRT. The key generation algorithm for Scheme A is very computationally expensive though. The second scheme, Scheme B, generates public exponents with $n_e<n/2$ and CRT exponents that are larger than that of Scheme A. Unlike the first scheme though, the key generation algorithm for Scheme B is very efficient. In addition to the proposed key generation algorithms we also presented a security analysis of Generalized Rebalanced RSA-CRT in terms of the parameters n_e and n_d that compliments the security analysis found in [13]. To ensure that the expected complexity of completely breaking an instance of Generalized Rebalanced RSA-CRT with an n-bit modulus is at least $O(2^m)$ each of the following inequalities should be satisfied:

$$5n_d + 2n_e > 2n + m,$$
$$3n_d + 2n_e > (3/2)n + m_1 \text{ or } n_e < n/4 - m_2,\ (m = m_1 + m_2)$$
$$6n_d + 3n_e > (5/2)n + m,$$
$$6n_d + 3n_e > (7/3)n - m - (1/2)(4m^2 - 4mn - 12mn_e + 6n_e n)^{0.5}, \text{ and } n_d > 2m.$$

Letting m=80 corresponds to the expected complexity of factoring a 1024-bit RSA modulus with the general number field sieve. It is an open question if stronger attacks on Rebalanced RSA-CRT exist.

Reference

[1] D. Boneh, G. Durfee and Y. Frankel, "*An Attacks on RSA Given a Small Fraction of the Private Key Bits*," Advanced in Cryptology--ASIACRYPT '98, LNCS 1514, Springer-Verlag, pp.25-34, 1998.

[2] D. Boneh, "*Twenty Years Attacks on the RSA Cryptosystem*," Notices of the American Mathematical Society, vol. 46:2, pp. 203-213, 1999.

[3] D. Boneh and G. Durfee, "*Cryptanalysis of RSA with private key d less than* $N^{0.292}$," IEEE Trans. on Information, Vol. 46(4), pp. 1339-1349, 2000.

[4] J. Blömer and A. May, "*Low Secret Exponent RSA Revisited*". CaLC , LNCS, vol. 2146 Springer-Verlag, pp.110-125, 2001.

[5] D. Boneh and H. Shacham, "*Fast Variants of RSA*," CryptoBytes, 2002, Vol. 5, No. 1, Springer, 2002.

[6] J. Blömer and A. May, "*New Partial Key Exposure Attacks on RSA*," Advanced in Cryptology--CRYPTO'03, LNCS 2729, Springer-Verlag, pp.27-43, 2003.

[7] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter, "*Low-Exponent RSA with related message*," Advances in Cryptology-EUROCRYPT'96, LNCS 1070, Springer-Verlag, pp.1-9, 1996.

[8] D. Coppersmith, "*Small solutions to polynomial equations, and low exponent RSA vulnerabilities*," Journal of Cryptology, vol. 10, pp.233-260, 1997.

[9] G. Durfee and P. Q. Nguyen, "*Cryptanalysis of the RSA Schemes with Short Secret Exponent form Asiacrypt '99*," Advances in Cryptology-Asiacrypt'00, LNCS 1976, Springer-Verlag, pp.1-11, 2000.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithm*, second edition, McGraw-hill Book Company, 2001.

[11] M. Ernst, E. Jochemsz, A. May and B. Weger, "*Partial Key Exposure Attacks on RSA Up to Full Size Exponents*," Advances in Cryptology-EUROCRYPT'05, LNCS 3494, Springer-Verlag, pp.371-386, 2005.

[12] N. Howgrave-Graham, "*Finding small roots of univariate modular equations revisited*," in Proceedings of Cryptography and Coding, LNCS, vol. 1355, Springer-Verlag, pp.131-142, 1997.

[13] S. D. Galbraith, C. Heneghan and J. F. McKee, "Tunable balancing of RSA," Information Security and Privacy: 10th Australasian Conference - ACISP 2005, LNCS 3574, pp. 280-292, 2005.

[14] http://www.alpertron.com.ar/ECM.HTM

[15] A. Lenstra, H. Lenstra, and L. Lovász, "*Factoring polynomials with rational coefficients*," Mathematische Annalen, vol. 261, pp.515-534, 1982.

[16] M. J. Hinek, "*Another Look at Small RSA Exponents*," to appear in Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006.

[17] A. May, "*Cryptanalysis of Unbalanced RSA with Small CRT-Exponent*," Advances in Cryptology-CRYPTO'02, LNCS 2442, pp.242-256, 2002.

[18] I. Niven and H. S. Zuckerman, *An Introduction to the Theory of Number*, John Wiley and Sons Inc,1991.

[19] J. J. Quisquater and C. Couvreur, "*Fast decipherment algorithm for RSA public key cryptosystem*," Electronic Letters, vol. 18, pp.905-907, 1982.

[20] G. Qiao and K.-Y. Lam, "*RSA Signature Algorithm for Microcontroller Implementation*," in proceeding of Smart Card Research and Applications, Third International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, LNCS, vol. 1820, Springer-Verlag, pp.353-356, 2000.

[21] R. Rivest, A. Shamir and L. Aldeman, "*A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*," Communications of the ACM , vol. 21, No.2, pp.120-126, 1978.

[22] V. Shoup, NTL: A Library for doing number theory, version 5.3.1. (Available online at http://shoup.net/ntl/)

[23] H.-M. Sun, W.-C. Yang and C.-S. Laih, "*On the Design of RSA with Short Secret exponent*," Advances in Cryptology-ASIACRYPT'99, LNCS 1716, pp.150-164, 1999.

[24] H.-M. Sun and C.-T. Yang, "*RSA with Balanced Short Exponents and Its Application to Entity Authentication*," Public Key Cryptography 05, 2005.

[25] H.-M. Sun and M.-E. Wu, "*Design of Rebalanced RSA-CRT for Fast Encryption*" Information Security Conference 2005. pp. 16-27, June, 2005.

[26] H.-M. Sun and M.-E. Wu, "*An approach Towards Rebalanced RSA-CRT with Short Public Exponent*," Cryptology ePrint Archive, 2005/053.

[27] E. Verheul and H. van Tilborg, "*Cryptanalysis of less short RSA secret exponents*," Applicable Algebra in Engineering, Communication and Computing, vol. 8, Springer-Verlag, pp. 425-435, 1997.

[28] M. J. Wiener, "*Cryptanalysis of RSA with short secret exponents*," IEEE Transactions on information Theory, IT-36, pp.553-558, 1990.

[29] M.-E. Wu, *A Study of RSA with Small CRT-Exponent*, Thesis of Master Degree, National Chiao Tung University, Taiwan, June 2004.

Appendix A: Examples of Scheme A and Scheme B

Appendix A.1: An example of Scheme A with $e=2^{591}+1$, d_p and d_q of 190 bits, p and q of 512 bits.

p = 1 0630FE3B E7B427CC 74299079 6B01213D 982856DB EC02470D 2BD74DA2
8BF1EF64 35CB21FB 3C3CD08A 2CDACFFC 080FB9C8 CC467AB0 28C62B58 6D74D170
5BEDE5BF

q = 1 2079FB3D C2DB286D 9FE50CD8 91ECCF9B 444C7169 6D71D98C 83C6A426
A6AC5FE2 1E48D500 4CD90E70 85C4EF87 8624C667 E11F217F 40AE6DCB 10F765D5
5EF7F4AF

d_p = 2ABFBC55 91531320 FD7C2818 30F64011 2CF3FEC8 282DCCDF

d_q = 36CDF94C F78366ED 5011A161 92307327 C496F08D 9DD98849

e = 8000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000001

Appendix A.2: An example of Scheme B with $e=2^{169}+1$, d_p of 359 bits, d_q of 357 bits, p of 512 bits and q of 511 bits.

p = C79380BD C277F1E2 AFAE9B95 8ECCDF7B 1CD7A75B BB6FBBB2 5019F1C0 1E826326
B1F29F40 92753A4A 635F1EC1 69BD7995 1DF04287 298179AB DDA83F46 2719B749

q = 66A23822 C2C9F41B 7F6039D5 E292FAE6 A7BA1B94 99BF159C 7E58F7C7 D6F9B5D8
3F4BA3C9 5B8FF4BD 4F7726FF C8D46F36 0ADCC154 FEBD650B 3ECD76CE 0FDB8CD1

d_p = 62 BF851AA4 190DF02B F1F25E3B 6471716F B0DA001C A7F489B8 5D11BD19
6285AAE5 BD3C401F DF6556EF 65AC02E9

d_q = 1E 8261D2E9 4A5B95FA 8DC14142 11B42077 C4F539DC E2C9C233 245A3E37
3CC4FB2F 33B9CE87 81790477 84588D71

e = 200 00000000 00000000 00000000 00000000 00000001

Appendix B: Scheme A$^+$

The most costly operations for any single operation in the key generation algorithm of Scheme A are the two factoring steps (4 & 7). Here we present another key generation algorithm for Rebalanced RSA-CRT, which we call Scheme A$^+$ that is about twice as fast as the key generation algorithm of Scheme A. Like Scheme A, the public exponents generated in Scheme A$^+$ all satisfy $n/2<n_e<n$. The main drawback to Scheme A$^+$, however, is that the public exponent cannot be chosen to be sparse. Thus, the encryption costs for Scheme A$^+$ are expected to be 1.5 times greater than the encryption costs in Scheme A for public exponents of the same size (n_e).

The key generation algorithm for Scheme A$^+$ takes (n, n_e, n_d) as input, with $n_e>n/2$, and outputs a valid public key (e, N) and corresponding private key (d_p, d_q, p, q), where $\log_2(N)=n$, $\log_2(e)=n_e$ and $\log_2(d_p)=\log_2(d_q)=n_d$. The algorithm is as follows:

Input: (n, n_e, n_d), where $n_e>n/2$.

Step 1. $p \leftarrow$ random (n/2)-bit prime, $d_p \leftarrow$ random (n_d)-bit odd integer such that gcd(d_p, p-1)=1.
Step 2. $k_{p1} \leftarrow$ random (n_e-n/2)-bit integer, using Theorem 4.1 (with h=2), compute (e, d_{p2}) satisfying $ed_p=k_{p1}k_{p2}$ (p-1)+1, where $k_{p1}(p\text{-}1)<e<2k_{p1}(p\text{-}1)$ and $d_p<k_{p2}<2d_p$.
Step 3. $k_{q1} \leftarrow$ random (n_d)-bit integer such that gcd (k_{q1}, e)=1.
Step 4. Using Theorem 4.1 (with h=2), compute (d_q, Q) satisfying $ed_q=k_{q1}Q+1$, where $k_{q1}<d_q<2k_{q1}$ and Q, $e<Q<2e$.
Step 5. Factor Q as $Q=k_{q2}(q\text{-}1)$ where k_{q2} is an (n_e-n/2)-bit number and q is prime. If this is infeasible, then go to Step 3.

Output: the public key (e, N) and the private key (d_p, d_q, p, q).

The encryption of Scheme A$^+$ is at least 1.7 times faster than that in the original Rebalanced RSA-CRT. Compared to Scheme A, Scheme A$^+$ provides much faster key generation but slower encryption time. Thus, we have a trade-off between encryption time and key generation time. The time of the key generation algorithms we have run is 912 seconds in average and it is actually near half of the time compared with Scheme A.