Dual RSA and Its Security Analysis

Hung-Min Sun, Mu-En Wu, Wei-Chi Ting, and M. Jason Hinek

***Abstract*—We present new variants of an RSA whose key generation algorithms output two distinct RSA key pairs having the same public and private exponents. This family of variants, called Dual RSA, can be used in scenarios that require two instances of RSA with the advantage of reducing the storage requirements for the keys. Two applications for Dual RSA, blind signatures and authentication/secrecy, are proposed. In addition, we also provide the security analysis of Dual RSA. Compared to normal RSA, the security boundary should be raised when applying Dual RSA to the types of Small-d, Small-e, and Rebalanced-RSA.**

***Index Terms*—Cryptography, encryption, lattice basis reduction, LLL algorithm, rebalanced RSA, RSA, twin RSA.**

I. Introduction

Since the mid-1970s, when public-key cryptography was first developed, the RSA Cryptosystem [33] has become the most popular cryptosystem in the world. Based on the believed difficulty of computing eth roots modulo N, where N is the product of two large unknown primes, it is widely believed to be secure for large enough N . Since RSA can also be broken by factoring N, the security of RSA is often based on the integer factorization problem, which is and continues to be a well-studied problem. Currently, it is suggested that the bitlength of N should be at least 1024 for RSA to be considered secure. Using the best known factoring algorithms, the expected workload of factoring a 1024-bit modulus is 2^{80} which is currently believed to be infeasible. A workload of 2^{80} is the current cryptographic benchmark used for security.

One of the reasons that RSA is so popular is its simplicity. Both encryption and decryption require only one modular exponentiation. However, computing an exponentiation modulo N is very costly because the RSA modulus is much larger than other moduli of public key cryptosystems such as those based on elliptic curves. The other main disadvantage of using RSA is the size of the key pairs. For example, to offer the same level of security as a given symmetric key cryptosystem, the RSA keysize must be much larger than that of elliptic curve based cryptosystems. In addition, as the security level is increased the RSA keysize grows at a much faster rate than the keys in an elliptic curve cryptosystem. For a detailed discussion about keysizes, see Lenstra [25].

In order to overcome these drawbacks, many researchers have studied variants of RSA which either reduce the computational costs [8], [19], [31], [37], [36], [38], [39], or reduce the (key) storage requirements [24], [26], [41].

In this work, we are interested in reducing the (key) storage requirement of RSA. In particular, we focus on the situation of using two RSA systems simultaneously. To this end, we introduce a variant of RSA, called *Dual RSA*, which consists of two distinct instances RSA that have the same public and private exponents. The memory needed to store both keys in Dual RSA is thus reduced since there is no need to store the same public/private exponent twice. Another variant of RSA that can be used to reduce the (key) storage requirement when two RSA systems are used is Twin RSA [26], proposed by Lenstra and de Weger.

The remainder of this work is organized as follows. In Section II , we review the RSA and some of its variants. In Section III, we present key generation algorithms for three variants of Dual RSA. In Section IV, we discuss two scenarios in which Dual RSA can be used. In Section V, we consider the security of Dual RSA. Finally, we end with Section VI, where we make some concluding remarks and comment on some open questions.

II. RSA and Its Variants

In this section, we review the RSA cryptosystem, as originally presented, and some of its variants. For a survey on fast variants of RSA see Boneh and Shacham [8] and Sun *et al.* [36]. In this work, we are mainly concerned with three variants: RSA-Small-e, RSA-Small-d, and Generalized Rebalanced-RSA.

We also mention Twin RSA, another variant of RSA that allows for reduced (key) storage when two instances of RSA are required.

A. Original RSA and Small Exponent RSA

The original RSA cryptosystem [33] consists of three algorithms: key generation, encryption, and decryption. Below, we describe each algorithm from the original description of RSA, sometimes called the textbook or simplified version of RSA, and then discuss some simple variants of the original presentation. In practice, an appropriate padding scheme, such as OAEP [2], is required to ensure the security of the cryptosystem.

Key Generation: Let $N = pq$ be the product of two randomly chosen large prime numbers p and q that are distinct. Let e be a randomly chosen integer that is relatively prime to $\varphi(N) = (p-1)(q-1)$, where $\varphi(\cdot)$ is Euler's phi function, and let d be its multiplicative inverse modulo $\varphi(N)$ (i.e., $ed \equiv 1 \pmod{\varphi(N)}$). The pair (e, N) is the public key and the pair (d, N) is the private key.

Encryption: A plaintext message $M \in \mathbb{Z}_N$ is encrypted by raising it to the eth power modulo N. The result, $C = M^e \bmod N \in \mathbb{Z}_N$, is called the ciphertext of M. All of the different variants of RSA in this correspondence use this method for encryption.

Decryption: A ciphertext $C \in \mathbb{Z}_N$, for a given plaintext message $M \in \mathbb{Z}_N$, is decrypted by raising it to the dth power modulo N. From Lagrange's theorem, it follows that

$$C^d \bmod N = M^{ed} \bmod N = M \bmod N = M.$$

The integer N is called the RSA modulus or simply the modulus. The integer e is called the public (or encryption) exponent and the integer d is called the private (or decryption) exponent. When computed in the manner described above, the private exponent d will, with high probability, be roughly the same order of magnitude as $\varphi(N)$. The public and private exponents are defined so that $ed \equiv 1 \pmod{\varphi(N)}$. We call this the RSA key relation, or simply the key relation. From the key relation, it follows that there exists a unique positive integer k satisfying

$$ed = 1 + k\varphi(N). \tag{1}$$

We call this the RSA key equation or simply the key equation.

Manuscript received September 28, 2006; revised January 15, 2007. This work was supported in part by the National Science Council, Taiwan, under Contract NSC 95-2221-E-007-030.

H.-M. Sun, M.-E. Wu, and W.-C. Ting are with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 300 (e-mail: hmsun@cs.nthu.edu.tw; mn@is.cs.nthu.edu.tw; sd@is.cs.nthu.edu.tw).

M. J. Hinek was with the David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: mjhinek@alumni.uwaterloo.ca).

Communicated by E. Okamoto, Associate Editor for Complexity and Cryptography.

Color version of Figure 1 is available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2007.901248

The main computational costs of RSA are the modular exponentiations found in the encryption and decryption operations. For a fixed modulus size a simple method to decrease these operations is to use small exponents. We consider RSA with one of the public or private exponents chosen significantly smaller than $\varphi(N)$ to be a fast variant of RSA.

RSA-Small-e: When the public exponent is much smaller than $\varphi(N)$ the encryption costs can be significantly reduced. The key generation algorithm can be trivially modified to generate instances of RSA with a public key of a specified size; the only change is to let e be a random integer of specified size satisfying $\gcd((p-1)(q-1), e) = 1$. Rather than wanting a random public exponent with a specific size it is often convenient to use a fixed value for the public exponent such as $e = 3$ or $e = 2^{16}+1$. Using public exponents this small is very desirable as it reduces encryption to only a few modular multiplications. However, in this case the key generation algorithm must be altered so that it is guaranteed to terminate. The following modification achieves this: given a specified public exponent e, let $N = pq$ be the product of two randomly chosen large distinct primes p and q such that $\gcd((p-1)(q-1), e) = 1$. The private exponent d is then computed using p, q and e in the same way.

The security of RSA-Small-e under various threat assumptions is considered by Coppersmith *et al.* [12], Coppersmith [11], and Boneh *et al.* [7]. When used properly (i.e., using a proper padding scheme, protecting the private key from side channel attacks, not sending related messages, etc.), RSA-Small-e is considered to be safe for public exponents as small as $e = 3$. The public key $e = 2^{16} + 1$ is the most popular choice in practice though.

Since the private exponent d in RSA-Small-e is always computed as the inverse of e modulo $\varphi(N)$, it is expected with high probability that d will be the same size as $\varphi(N)$. So, while the encryption costs are reduced in this variant, the decryption costs remain the same as for original RSA.

RSA-Small-d: When the private exponent is much smaller than $\varphi(N)$ the decryption costs can be significantly reduced. Generating instances of RSA with a small private exponent is easy with the observation that the key relation (and equation) are symmetric with respect to the public and private exponent. To generate an instance of Small-RSA-d we simply follow the same key generation for RSA-Small-e and exchange the public and private exponents (i.e., swap the values of e and d).

Unlike the small public exponent scenario RSA-Small-e, it is not safe to use very small private exponents. In particular, Boneh and Durfee [6] have shown that any private exponent $d < N^{0.292}$ should be considered unsafe.

Since the public exponent e in RSA-Small-d is always computed as the inverse of d modulo $\varphi(N)$, it is expected with high probability that e will be the same size as $\varphi(N)$. Thus, in this variant the decryption costs are reduced while the encryption costs remain the same as for original RSA.

In each of the small exponent variants above one of the exponents is much smaller than $\varphi(N)$ while the other is, with high probability, the same size as $\varphi(N)$. Thus, either the encryption costs or the decryption costs, but not both, can be decreased. The choice of which variant to employ then depends on which operation cost (encryption or decryption) needs to be decreased. It is also possible to generate instances of RSA in which both the public and private exponents are much smaller than $\varphi(N)$. For more information see Sun and Yang [38].

B. CRT-Decryption

In practice, the RSA decryption computations are performed in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ and then combined via the Chinese Remainder Theorem (CRT) to obtain the desired solution in $\mathbb{Z}_N$, instead of directly computing the exponentiation in $\mathbb{Z}_N$. This decreases the computational costs of decryption in two ways. First, computations in $\mathbb{Z}_p$ and $\mathbb{Z}_q$ are more efficient than the same computations in $\mathbb{Z}_N$ since the elements are much smaller. Second, from Lagrange's Theorem, we can replace the private exponent d with $d_p = d \bmod (p-1)$ for the computation in $\mathbb{Z}_p$ and with $d_q = d \bmod (q-1)$ for the computation in $\mathbb{Z}_q$, which reduce the cost for each exponentiation when d is larger than the primes. It is common to refer to d_p and d_q as the CRT-exponents. The first method to use the CRT for decryption was proposed by Quisquater and Couvreur [32].

Since the method requires knowledge of p and q, the key generation algorithm needs to be modified to output the private key (d, p, q) instead of (d, N). Given the private key (d, p, q) and a valid ciphertext $C \in \mathbb{Z}_N$, the CRT-decryption algorithm is as follows:

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

This version of CRT-decryption is simply Garner's Algorithm for the Chinese Remainder Theorem applied to RSA. If the key generation algorithm is further modified to output the private key $(d_p, d_q, p, q, p^{-1} \bmod q)$, the computational cost of CRT-decryption is dominated by the modular exponentiations in steps 1) and 2) of the algorithm. When the primes p and q are roughly the same size (i.e., half the size of the modulus), the computational cost for decryption using CRT-decryption (without parallelism) is theoretically $1/4$ the cost for decryption using the original method.

Using RSA-Small-e along with CRT-decryption allows for extremely fast encryption and decryption that is at most four times faster than standard RSA.

C. Rebalanced-RSA

In some situations it is desirable to reduce decryption costs as much as possible (see, for example, [8, Sec. 4]. Rebalanced-RSA is a variant, proposed by Wiener [44], that accomplishes this by shifting the cost of decryption to encryption. Essentially, one chooses a private exponent d so that the CRT-exponents, d_p and d_q, are small.

The following key generation algorithm, taken from [8], will compute an instance of Rebalanced-RSA with an n-bit modulus and ℓ-bit CRT-exponents:

1) Randomly select two distinct $(n/2)$-bit primes $p = 2p_1 + 1$ and $q = 2q_1 + 1$ such that $\gcd(p_1, q_1) = 1$. Let $N = pq$.
2) Randomly select two ℓ-bit integers d_p and d_q such that $\gcd(d_p, p-1) = 1$, $\gcd(q-1, d_q) = 1$ and $d_p \equiv d_q \pmod 2$.
3) Compute d such that $d \equiv d_p \pmod{(p-1)}$ and $d \equiv d_q \pmod{(q-1)}$.
   a) Let $a = d_p \bmod 2 = d_q \bmod 2$.
   b) Using the CRT, compute d' such that $d' \equiv \frac{d_p - a}{2} \pmod{(\frac{p-1}{2})}$ and $d' \equiv \frac{d_q - a}{2} \pmod{(\frac{q-1}{2})}$.
   c) Let $d = 2d' + a$.
4) Compute $e = d^{-1} \bmod \varphi(N)$

The public key is (e, N) and the private key is (d_p, d_q, p, q). The requirement that $d_p \equiv d_q \pmod 2$ in step 2) is necessary to allow the determination of d in step 3. The structure of d (from step 3) implies that there exist positive integers k_p and k_q such that

$$\begin{aligned} e d_p &= 1 + k_p(p-1) \\ e d_q &= 1 + k_q(q-1). \end{aligned} \tag{2}$$

We call these equations the RSA-CRT equations.

Since the public exponent e is computed as the inverse of d modulo $\varphi(N)$, it is expected that the e will have roughly the the same order

of magnitude as $\varphi(N)$. Thus, the decryption costs have been lowered at the expense of maximizing the encryption costs (i.e., the encryption costs are the same as original RSA). While the CRT-exponents can be chosen smaller than $N^{0.292}$ (the limit for RSA-Small-d), they cannot be chosen arbitrarily small. Notice that the knowledge of one CRT-exponent allows to factor N in probabilistic polynomial time by computing $\gcd(m^{ed_p}-m, N)$ for random m [5]. Thus Baby-Step Giant-Step proposed by Qiao and Lam [31] can be applied to factor N and the attack requires the complexity $O(\min\{\sqrt{d_p}, \sqrt{d_q}\})$. For a workload of 2^{80} steps, which is essentially the work required to factor a 1024-bit RSA modulus with NFS, the CRT-exponents should each be at least 160 bits.

D. Generalized Rebalanced-RSA

New key generation algorithms for Rebalanced-RSA, independently proposed by Galbraith *et al.* [14] and Sun and Wu [37], allow for instances of Rebalanced-RSA with small public exponent and small CRT-exponents. We call such instances Generalized Rebalanced-RSA.

The key generation algorithms for Generalized Rebalanced-RSA differ from Rebalanced-RSA in that the private exponent d is never explicitly computed and the primes p and q are generated at the end of the algorithm (as opposed to the start). Since the key generation algorithms and security analysis are more complex (and space consuming) [3] , we refer the reader to Galbraith *et al.* [15] and Sun *et al.* [36], (the most recent versions of their works) for more detail.

We end our discussion about Generalized Rebalanced-RSA by mentioning that knowledge of d_p and d_q allow the construction of an equivalent decryption exponent. Multiplying together the CRT-key (2), written as $\{ed_p - 1 = k_p(p-1), ed_q - 1 = k_q(p-1)\}$, yields after some rearrangement

$$\begin{aligned} e(-ed_pd_q + d_p + d_q) &= 1 - k_pk_q(p-1)(q-1) \\ &= 1 - k_pk_q\varphi(N). \end{aligned}$$

Letting $D = -ed_pd_q + d_p + d_q$, it follows that e and D satisfy the key relation $eD \equiv 1 \pmod{\varphi(N)}$. Therefore, D can be used to decrypt any ciphertext generated by the public exponent e (with modulus N). The construction of D also applies to Rebalanced-RSA.

E. Twin RSA

Twin RSA [26] is a variant of RSA which consists of pairs of RSA moduli $(N, N+\Delta)$, where Δ is a small even integer such as $\Delta = \pm 2$. The advantage of this system is that, once Δ is fixed, there is no need to store both moduli since N and Δ specify the second modulus. Thus, in situations that require two instances of RSA, Twin RSA can be used to reduce the (key) storage requirements. Since Δ is small, the storage requirements is reduced by the size of one RSA modulus.

The algorithms presented by Lenstra and de Weger [26] for Twin RSA generate pairs of RSA moduli $(N, N+\Delta)$. Once the moduli are computed, the public and private (or CRT-) exponents are computed. As a result, it is obvious that Small-e and Small-d variants of Twin RSA are possible. However, since all known algorithms for Generalized Rebalanced-RSA compute the exponents in conjunction with the RSA primes, it is unknown if a Generalized Rebalanced-Twin-RSA variant is possible.

For more detail of Twin RSA, such as moduli generation algorithms and security, see [26].

F. Some Notation and Assumptions

In addition to the notation laid out previously, we use the following notation and assumptions throughout the remainder of this correspondence.

We only consider instances of RSA with balanced primes. That is, when the RSA primes satisfy $1/2 < p/q < 2$. When the RSA primes are balanced, the difference between N and $\varphi(N)$ is quite small (compared to the size of N). In particular, $N - \varphi(N) = p+q-1 < 3N^{1/2}$. To simplify notation we often use s to denote the difference between N and $\phi(N)$ (i.e., $s = p + q - 1$).

We let $|x|$ denote the bitlength of any $x \in \mathbb{N}$. Thus, we have $2^{|x|-1} \leqq x < 2^{|x|}$.

III. Dual RSA

Dual RSA is essentially two distinct instances of RSA that share the same public and private exponents. Combining the two instances we obtain one Dual RSA instance with public key (e, N_1, N_2) and private key (d, p_1, q_1, p_2, q_2), where e and d satisfy $ed \equiv 1 \pmod{\varphi(N_1)}$ and $ed \equiv 1 \pmod{\varphi(N_2)}$. From these two relations, it follows that there exists two positive integer k_1 and k_2 such that

$$\begin{aligned} ed &= 1 + k_1\varphi(N_1) \\ ed &= 1 + k_2\varphi(N_2) \end{aligned} \tag{3}$$

which we call the Dual RSA key equations or simply the key equations. The main idea behind the key generation algorithms that we present for Dual RSA comes from the equation $k_1\varphi(N_1) = k_2\varphi(N_2)$, which directly follows from the key (3). The idea is to construct three integers k_1, k_2, and k_3 such that $k_2k_3 = (p_1-1)(q_1-1)$ and $k_1k_3 = (p_2-1)(q_2-1)$, where p_1, q_1, p_2 and q_2 are all primes.

When small CRT-exponents are used, such as in Rebalanced-RSA and Generalized Rebalanced-RSA, the Dual version has public key (e, N_1, N_2) and private key $(d_p, d_q, p_1, q_1, p_2, q_2)$, where e, d_p and d_q satisfy $ed_p \equiv 1 \pmod{(p_i - 1)}$ and $ed_q \equiv 1 \pmod{(q_i - 1)}$ for $i = 1, 2$. From these four relations, it follows that there exists four positive integers k_{p_1},k_{q_1}, k_{p_2} and k_{q_2} such that

$$\underbrace{\begin{aligned} ed_p &= 1 + k_{p_1}(p_1 - 1) \\ ed_q &= 1 + k_{q_1}(q_1 - 1) \end{aligned}}_{\text{for } N_1} \quad , \quad \underbrace{\begin{aligned} ed_p &= 1 + k_{p_2}(p_2 - 1) \\ ed_q &= 1 + k_{q_2}(q_2 - 1) \end{aligned}}_{\text{for } N_2} \tag{4}$$

which we call the Dual RSA-CRT equations, or simply the CRT equations. The main idea behind the key generation algorithm for Dual RSA with small CRT-exponents comes from the equations $k_{p_1}(p_1 - 1) = k_{p_2}(p_2 - 1)$ and $k_{q_1}(q_1 - 1) = k_{q_2}(q_2 - 1)$, which are obtained from (4). The idea is essentially the same as described above except that we now need to satisfy two equations when finding the primes p_1, q_1, p_2 and q_2.

Below we present the key generation algorithms for three variants of Dual RSA: Dual RSA-Small-e, Dual RSA-Small-d and Dual Generalized Rebalanced-RSA. The encryption and decryption algorithms are the same for each scheme and correspond to what is done, in practice, with RSA. That is, encryption follows the standard method and decryption is done using the Chinese Remainder Theorem.

A. Dual RSA-Small-e (Scheme I)

The first scheme we consider is Dual RSA with small public exponent, which we call Dual RSA-Small-e. The key generation algorithm takes (n_e, n) as input (with $n_e < n/2$) and outputs a valid public/private key pair with an n_e-bit public exponent and two n-bit moduli. The Dual RSA-Small-e key generation algorithm, with (n_e, n) as input, is given in Algorithm 1.

Algorithm 1: KEY-GEN-SCHEME-I

Input: (n_e, n) such that $n_e < n/2$.

1) Randomly select an n_e-bit integer x_1 and an $(n/2 - n_e)$-bit integer x_2 such that $p_1 = x_1x_2 + 1$ is prime.
2) Randomly select an $(n/2 - n_e)$-bit integer y_2 such that $p_2 = x_1y_2 + 1$ is prime.
3) Randomly select an n_e-bit integer y_1 such that $q_1 = y_1y_2 + 1$ is prime.

4) Randomly select an n_e-bit integer e such that $\gcd(x_1x_2y_1y_2, e) = 1$. Compute d and k_1 satisfying $ed = 1 + k_1(p_1 - 1)(q_1 - 1)$.
5) If $q_2 = k_1x_2 + 1$ is not prime then go back to step 4).
6) Let $N_1 = p_1q_1$, $N_2 = p_2q_2$, and $k_2 = y_1$.

Output: (e, N_1, N_2) and (d, p_1, q_1, p_2, q_2). ◇

The correctness of the key generation algorithm follows since

$$\begin{aligned} ed &= 1 + k_1\varphi(N_1) = 1 + k_1(p_1 - 1)(q_1 - 1) \\ &= 1 + k_1(x_1x_2)(y_1y_2) = 1 + y_1(x_1y_2)(k_1x_2) \\ &= 1 + k_2(p_2 - 1)(q_2 - 1) = 1 + k_2\varphi(N_2). \end{aligned} \tag{5}$$

Therefore, e and d are a valid public/private exponent pair for both N_1 and N_2 since $ed \equiv 1 \pmod{\varphi(N_1)}$ and $ed \equiv 1 \pmod{\varphi(N_2)}$.

The key generation algorithm outputs a random n_e-bit public exponent and, with high probability, an n-bit private exponent. In some situations a specific public exponent might be desired though. For example, public exponents $e = 2^m + 1$ that have low Hamming weight are often used to reduce the cost of square-and-multiply algorithms for modular exponentiation. The key generation algorithm can be easily modified to generate instances of Dual RSA-Small-e with a fixed public exponent. For a specified n_e-bit public exponent e the modified algorithm, taking as input (e, n_e, n), is given in the Appendix

In standard RSA, public exponents as small as $e = 3$ are considered safe when used properly. In Section V-B, we show that this is not the case for Dual RSA. In particular, Dual RSA is insecure when public exponents smaller than $N^{1/4}$ are used.

B. Dual RSA-Small-d (Scheme II)

The second scheme we consider is Dual RSA with small private exponent, which we call Dual RSA-Small-d. The key generation algorithm takes (n_d, n) as input (with $n_d < n/2$) and outputs a valid public/private key pair with an n_d-bit private exponent and two n-bit moduli.

The key generation algorithm for Dual RSA-Small-d is exactly the same as the key generation for Dual RSA-Small-e, except replacing e, n_e and d with d, n_d and e, respectively. This follows from the symmetry of e and d in the key equations (3). Thus, the correctness and efficiency of generating Dual RSA instances with small private exponents are the same as shown in the previous section for Dual RSA-Small-e.

In standard RSA, private exponents smaller than $N^{0.292}$ are considered unsafe. In Section V.C, we show that this bound increases to $N^{0.333}$ for Dual RSA-Small-d.

C. Dual Generalized Rebalanced-RSA (Scheme III)

The last scheme, we consider is Dual RSA with small public exponent and small CRT-exponents, which we call Dual Generalized Rebalanced-RSA. As the name suggests, this scheme is an extension of Generalized Rebalanced-RSA to the Dual RSA setting. Thus, the public exponent e and CRT-exponents d_p and d_q should be valid exponents for two moduli ($N_1 = p_1q_1$ and $N_2 = p_2q_2$).

The key generation algorithm we present takes (n_e, n_d, n_k, n) as input (with $n_e < n/2$ and $n_e + n_d = n/2 + n_k$) and outputs a valid public/private key pair with an n_e-bit public exponent, an n_d-bit private exponent and n-bit moduli. The value n_k is a security parameter which is the bitlength of the constants k_{p_i} and k_{q_i} (for $i = 1, 2$) from (4). The following result from number theory (see [17] or [28]) will be used in the key generation algorithm.

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

The integers (u_h, v_h) can be efficiently computed using the Extended Euclidean Algorithm. We use the result of Theorem 1 in the key generation algorithm to ensure the existence of certain numbers with a given bitlength. Finally, the Dual Generalized Rebalanced-RSA key generation algorithm, with (n_e, n_d, n_k, n) as input, is given in Algorithm 2.

Algorithm 2: KEY-GEN-SCHEME-III
Input: (n_e, n_d, n_k, n).
Pre: $n_e < n/2$ and $n_e + n_d = n/2 + n_k$.
1) Randomly select an n_e-bit integer e and let k be the smallest integer larger than $(n/2 - n_e)/n_k$ (i.e., $k = \lceil (n/2 - n_e)/n_k \rceil$).
2) Randomly select $k-1$ n_k-bit integers $p_{a_1}, \ldots, p_{a_{(k-1)}}$ and an even integer p_{a_k} such that $p_a = p_{a_1} \cdots p_{a_{(k-1)}} p_{a_k}$ has bitlength $(n/2 - n_e)$ and $\gcd(e, p_a) = 1$.
3) Randomly select an n_k-bit integer k_{p_1} such that $\gcd(e, k_{p_1}) = 1$.
4) Based on Theorem 1, compute d_p and p_b such that $ed_p = (k_{p_1}p_a)p_b + 1$, where $e < p_b < 2e$ and $k_{p_1}p_a < d_p < 2k_{p_1}p_a$. If $p_1 = p_ap_b + 1$ is not prime then go to step 3).
5) If $(k_{p_1}p_ap_b/p_{a_{i'}}) + 1$ is prime for some $1 \le i' \le k-1$ then let $p_2 = (k_{p_1}p_ap_b/p_{a_{i'}}) + 1$. Otherwise, go to step 3).
6) Randomly select $k-1$ n_k-bit integers $q_{a_1}, \ldots, q_{a_{(k-1)}}$ and an even integer q_{a_k} such that $q_a = q_{a_1} \cdots q_{a_{(k-1)}} q_{a_k}$ has bitlength $(n/2 - n_e)$ and $\gcd(e, q_a) = 1$.
7) Randomly select an n_k-bit integer k_{q_1} such that $\gcd(e, k_{q_1}) = 1$.
8) Based on Theorem 1, compute d_q and q_b such that $ed_q = (k_{q_1}q_a)q_b + 1$, where $e < q_b < 2e$ and $k_{q_1}q_a < d_q < 2k_{q_1}q_a$ If $q_1 = q_aq_b + 1$ is not prime then go to step 7).
9) If $(k_{q_1}q_aq_b/q_{a_{j'}}) + 1$ is prime for some $1 \le j' \le k-1$ then let $q_2 = (k_{q_1}q_aq_b/q_{a_{j'}}) + 1$. Otherwise, go to step 7).
10) Let $N_1 = p_1q_1$, $N_2 = p_2q_2$, $k_{p_2} = p_{a_{i'}}$ and $k_{q_2} = q_{a_{j'}}$.

Output: (e, N_1, N_2) and $(d_p, d_q, p_1, q_1, p_2, q_2)$. ◇

The correctness of the algorithm follows since

$$\begin{aligned} ed_p &= 1 + k_{p_1}(p_1 - 1) \\ &= 1 + k_{p_1}(p_ap_b) \\ &= 1 + k_{p_1}(p_{a_1} \cdots p_{a_{i'}} \cdots p_{a_k}p_b) \\ &= 1 + p_{a_{i'}}(p_{a_1} \cdots k_{p_1} \cdots p_{a_k}p_b) \\ &= 1 + k_{p_2}(p_2 - 1), \end{aligned} \tag{6}$$

for some $i' \in \{1, \ldots, k-1\}$. Similarly, $ed_q = 1 + k_{q_1}(q_1 - 1) = 1 + k_{q_2}(q_2 - 1)$ for some $j' \in \{1, \ldots, k-1\}$ such that $k_{q_2} = q_{a_{j'}}$. Therefore, the CRT equations (4) are satisfied and e, d_p and d_q are a valid public/CRT-exponent pair for both N_1 and N_2.

From the key generation algorithm of scheme III, notice that $p_1 - 1$, $p_2 - 1$, $q_1 - 1$ and $q_2 - 1$ each has some small factors. This follows since p_a and q_a each has k factors (not necessarily prime) which are equal to or smaller than $n_k (= n_e + n_d - n/2)$ bits long (i.e., p_a and q_a are $2^{n_k - 1}$-smooth). This also implies that the primes p_1, q_1, p_2, and q_2 in the Dual Generalized Rebalanced-RSA moduli must not be strong primes.

Fortunately, the requirement (or suggestion) that RSA primes should be strong primes is no longer needed due to [16], [20], [34], and [35]. In fact, Rivest and Silverman [34], concluded that the development of elliptic curve method (ECM) [23] for factoring demotes any special status that attacks such as Pollard's $p - 1$ method may have had. Of course, this assumes that the primes are randomly generated and since the primes generated in Algorithm 2 have significant structure imposed on them, it is clear that they are not randomly sampled from the set of all $n/2$-bit primes. Therefore, care must be taken that the special nature of the primes generated in Algorithm 2 do not lend themselves easily to a specific factoring attack. In particular, if p_b or q_b is sufficiently

TABLE I
PRACTICAL EFFICIENCY OF ALGORITHM 1

1024-bit moduli		2048-bit moduli	
n_e	#(rand nums)	n_e	#(rand nums)
250	1283	500	2424
300	1214	600	2373
350	1277	700	2558
400	1264	800	2453
450	1242	900	2561
500	1228	1000	2448
average	1251	average	2466

TABLE II
PRACTICAL EFFICIENCY OF ALGORITHM 2

1024-bit moduli		2048-bit moduli	
n_e, n_d	#(rand nums)	n_e, n_d	#(rand nums)
$170, 509$	41303	$300, 1019$	144885
$213, 466$	72802	$405, 914$	171122
$256, 423$	86716	$510, 809$	330972
$299, 380$	70881	$615, 704$	288689
$342, 337$	95130	$720, 599$	365333
average	73366	average	260200

smooth, then Pollard's $p-1$ method can be used to factor one of the moduli. For example, if all the prime-factors of $p_1 - 1$ are smaller than B then Pollard's $p-1$ method can factor N_1 in $B \times \log B \times \log^2 N_1$ operations. Since p_b and q_b are each n_e-bit numbers, and there is no apparent structure to their prime composition from their construction in Algorithm 2, we can assume that each will have a prime-factor that is at least $n_e/2$ bits long with high probability. Consequently, Pollard's $p-1$ method is infeasible for factoring our moduli while using a sufficiently large public exponent. For example, for 1024-bit moduli and public exponent satisfying $n_e > 100$, we expect with high probability that p_b and q_b will each have a prime-factor larger than $n_e/2 > 50$ bits long. Thus, the required operations of Pollard's $p-1$ method will be expected to be at least 2^{80} times and the moduli will not be vulnerable to this attack.

The parameter selection for Dual Generalized Rebalanced-RSA is discussed further in Section V.D.

D. Efficiency of the Key Generation Algorithms

Given the correctness of the key generation algorithms, Algorithm 1 and 2, we need to consider their efficiency. For each algorithm we base the complexity on the total number of random numbers that need to be sampled in order to successfully generate a valid key pair. For each random number sampled, both algorithms perform some number of computations whose complexities are at most polynomial in the size of the moduli (i.e., n).

In Algorithm 1, steps 1)–4) are essentially sampling random numbers of specified bitlengths until the four primes (p_1, q_1, p_2, q_2) are found satisfying the key equations for some e and d. From the Prime Number Theorem, we estimate that the expected number of random numbers needed to generate a valid key pair with n-bit moduli is $O(n)$. To demonstrate that this complexity is meaningful for finite moduli sizes we counted the number of random numbers required to generate valid key pairs with 1024-bit and 2048-bit moduli. The results are given in Table I, where we show the average taken over 1000 key pairs for each public key size. The total number of random numbers needed, averaged over all keys generated, was 1251 for 1024-bit moduli and 2466 for 2048-bit moduli. The average time[1] needed to generate a valid Dual RSA key pair, averaged over all keys generated, was 18.5 s for 1024-bit moduli and 356.7 s for 2048-bit moduli. Notice that the average number of random numbers needed to generate a key pair with 2048-bit moduli compared to the average number needed to generate a key pair with 1024-bit moduli is 1.97, which is quite close to 2. This is in close agreement with the theoretical estimation that the number of random numbers needed increases linearly with the bitsize of the moduli.

In Algorithm 2, we again sample many random numbers of specified bitlengths until four primes, (p_1, q_1, p_2, q_2), are found satisfying the CRT equations (4) for some e, d_p and d_q. Of course, the conditions are more strict in this algorithm and so we expect the average number of random numbers needed to be larger than the previous algorithm. In Table II, we show the average number of random numbers needed to generate keys with 1024-bit and 2048-bit moduli for various combinations of n_e and n_d. The averages are based on 50 trials for each combination. As expected, the average number of random numbers needed is larger for this algorithm compared to the previous one. In fact, the average is roughly two orders of magnitude greater. The average number of random numbers needed for 2048-bit moduli is roughly three to four times that needed for 1024-bit moduli.

[1]The timing measurements for both algorithms were performed using a Java implementation on a Sun Fire V100 server with one UltraSPARC IIe processor with 2 GB of memory running at 550 MHz.

IV. APPLICATIONS

In this section we consider scenarios in which Dual RSA can be used. Essentially, whenever two RSA key pairs are required, Dual RSA can be used to decrease the storage requirements. We consider two applications: blind signatures and authentication/secrecy.

A. Blind Signatures

The first application we consider is blind signatures. The concept of blind signatures was first introduced by Chaum [9]. Essentially, it allows one user to have a message signed by another user without revealing any information about the message to the signer. There are many possible applications for blind signatures such as e-cash, untraceable electronic mail, electronic election systems, time-stamping, and anonymous access control. For a bibliography of blind signatures, see Wang [43].

We are concerned with blind signatures based on RSA. Suppose Alice has a message m that she wishes to have signed by Bob but doesn't want Bob to know anything about. Let (e, N) be Bob's public key and (d, N) be his private key. Alice generates a random value r such that $\gcd(r, N) = 1$ and sends

$$x = r^e m \bmod N$$

to Bob. Notice that Bob cannot derive any useful information about m from x. The message m is said to be *blinded* by the random value r. Bob then returns the signed value

$$t = x^d \bmod N$$

to Alice. Since $x^d \equiv (r^e m)^d \equiv r m^d \pmod N$, Alice can obtain the true signature s of m by computing

$$s = r^{-1} t \bmod N = m^d \bmod N.$$

Alice now has a message m and a signature of m that is signed with Bob's private key.

When using RSA for blind signatures, however, it is important that Bob uses one key pair exclusively for blind signatures. If Bob uses one key pair for both encryption/decryption and signing blind signatures, a person-in-the-middle attack exists which can be used to decrypt encrypted messages sent to Bob. To see this, suppose Alice, wanting to

send an encrypted message m to Bob, uses Bob's public key (e, N) and sends

$$c = m^e \bmod N$$

to Bob. Eve intercepts the ciphertext c and sends

$$c_0 = r^e c \bmod N$$

to Bob asking for a blind signature. Bob computes the signature of c_0 with his private key (d, N) and sends

$$m_0 = c_0^d \bmod N$$

back to Eve. Finally, Eve can compute Alice's message m since

$$m = r^{-1} m_0 \bmod N.$$

Thus, Bob should not use the same RSA key pair for both encryption/decryption and signing blind signatures.

Therefore, any user wishing to use RSA for both encryption/decryption and generating blind signatures should have a different key pair for each usage. Using Dual RSA or Twin RSA in this scenario will thus allow the user to reduce the amount of information needed to be stored for the key pairs.

B. Authentication/Secrecy

The second application we consider is authentication/secrecy. In particular, we consider solutions to the reblocking problem of RSA signatures. The reblocking problem arises when RSA is used to first sign and then encrypt a message to ensure both authenticity and secrecy of the message.

Suppose Alice and Bob know each other's public keys. Alice wishes to send a message m to Bob such that Bob, upon receiving m, is confident that Alice sent the message (authentication) and Alice is confident that only Bob can read the message m (secrecy). To accomplish this, Alice computes c by first signing the message with her private key (d_A, N_A) and then encrypting the signature with Bob's public key (e_B, N_B). She then sends

$$c = (m^{d_A} \bmod N_A)^{e_B} \bmod N_B$$

to Bob. Since c is encrypted with Bob's public key, Alice is confident that only Bob can decrypt the ciphertext. After decrypting the ciphertext with his private key, Bob then encrypts the result with Alice's public key. If the final result is something meaningful, then Bob can be confident that only Alice could have created it since only she has knowledge of her private key. Thus, the transmission ensures both authenticity and secrecy. However, Bob is not guaranteed to recover the original message m (or anything meaningful at all) if Alice's RSA modulus N_A is greater than his RSA modulus N_B. In this case, when Bob decrypts the ciphertext he obtains $(m^{d_A} \bmod N_A) \bmod N_B$ which might be different than $(m^{d_A} \bmod N_A)$. This is the reblocking problem as originally identified by Rivest, Shamir and Adleman [33, Sec. X]. When $N_A > N_B$, the probability that Bob cannot recover the original message is $(N_A - N_B)/N_A$.

We discuss two solutions to the reblocking problem as given in [27, Sec. 11.3.3]: reordering and using two moduli per user.

1) Reordering: The first solution to the reblocking problem is to simply reorder the signing and encrypting operations depending on which RSA moduli is larger. Thus Alice first signs and then encrypts if $N_A < N_B$ or encrypts and then signs is $N_A > N_B$. This reordering solution is sometimes called the *pre-judgement method*. This method is undesirable for two reasons though. First, when Alice encrypts then signs, any observer can remove the signature with Alice's public key and replace it with their own. Without proper protocols, this may lead to a simple person-in-the-middle attack: Eve intercepts the transmitted message, removes Alice's signature, adds her own signature, sends to Bob and later asks Bob to send the message m back. A second undesirable property of encrypting before signing is when Alice wishes to send the same message to a group of users $\{B_1, B_2, \ldots, B_n\}$, each with public key (e_i, N_i). When $N_A < N_i$ for all $i = 1, \ldots, n$ then Alice only needs to sign the message once and then perform one encryption for each of the B_i. For each modulus smaller than N_A, however, Alice needs to compute an additional signature. In the worst case, when $N_A > N_i$ for all i, Alice needs to compute n encryptions and n signatures. Thus the total number of encryptions/signatures in the worst case $(2n)$ is almost twice as many as the best case $(n + 1)$.

2) Two Moduli Per User: A second solution to the reblocking problem was suggested by Rivest, Shamir, and Adleman [33, Sec. X]. The solution consists of each user having two RSA key pairs such that one modulus is smaller than some threshold, say h, and one modulus is larger. To send a message with authenticity/secrecy, Alice will first sign the message with her private key with modulus smaller than h and then encrypt with Bob's public key with modulus larger than h. In this way, the modulus used for signing is always smaller than the modulus used for encryption and Bob is always able to recover the original message. Additionally, using this method Alice can send the same message to a group of n users (as described above) with only one signature and n encryptions, which is the minimum number of encryptions/signatures required. We call this method the *threshold technique for authentication/secrecy*. In this method, two RSA key pairs are required for each user. Therefore, Dual RSA can be used with this method to reduce the amount of information needed to be stored for the keys.

V. Security Analysis

In this section we provide the security analysis of Dual RSA. For each scheme we review the known attacks on RSA and present new attacks specific to Dual RSA. Based on the attacks presented we then illustrate which parameters are insecure and suggest parameters that are secure (i.e., , are resistant to all known attacks). We first give a brief overview of the mathematical tools needed for the new attacks.

A. Mathematical Tools

There are two main mathematical tools that we use in the proposed attacks against Dual RSA: continued fractions and lattices. From continued fraction theory we make use of the following well known result (see, for example, [17]).

Theorem 2: Let a, b, c, and d be integers satisfying $|\frac{a}{b} - \frac{c}{d}| < \frac{1}{2d^2}$. Then $\frac{c}{d}$ is one of the convergents in the continued fraction expansion of $\frac{a}{b}$. $\diamond$

From lattice theory we use extensions of Coppersmith's methods for finding small solutions of polynomials and also use a common heuristic about small vectors in lattices. Before stating the results needed for the attacks we first give a brief review of lattice theory. For our purpose we will only consider full dimensional integer lattices. Let $\mathcal{B} = \{b_1, \ldots, b_m\}$ be a set of m linearly independent elements of $\mathbb{Z}^n$. The set $\mathcal{L} = \{\sum_{i=1}^{m} \alpha_i b_i | \alpha_i \in \mathbb{Z}\}$, is called an integer lattice, or simply lattice, spanned by $\mathcal{B}$. The set $\mathcal{B}$ is called a basis of $\mathcal{L}$ and the b_i are called basis vectors. The dimension of $\mathcal{L}$ is simply the number of vectors making up the basis, (i.e., m). Lattices with dimension $m \geq 2$ have infinitely many bases. The determinant of $\mathcal{L}$, denoted by $\det(\mathcal{L})$, is equal to the m-dimensional volume of the parallelepiped spanned by the basis vectors b_i and is independent of the choice of basis. When $\mathcal{L}$ is full dimensional (i.e., $n = m$) $\det(L)$ is simply the absolute value of

the determinant of $\mathcal{B}$, where we consider the basis $\mathcal{B}$ as a matrix (whose rows or columns are the basis vectors).

Often we are interested in a smallest vector of a lattice. The following result by Minkowski gives an upper bound on the size of a smallest vector in a lattice as a function of the determinant and dimension of the lattice.

Theorem (Minkowski): Let $\mathcal{L}$ be an n-dimensional lattice. There exists $v \in \mathcal{L}$ such that

$$\|v\| \leq n^{1/2} \det(\mathcal{L})^{1/n}$$

where $\|\cdot\|$ denotes the 2-norm. $\diamond$

Given a basis for a lattice, Lovász's lattice basis reduction algorithm, usually referred to as the Lenstra–Lenstra–Lovász (LLL) algorithm (see [22]), can be used to find reasonably small vectors in lattices in polynomial time. The following result gives an upper bound on the size of the smallest vector computed by the LLL algorithm as a function of the determinant and dimension of the lattice (*cf.* Theorem 3).

Theorem 4 (LLL): Let $\mathcal{B} = \{b_1, \ldots, b_n\}$ be a basis of a lattice $\mathcal{L}$ with dimension n. On input $\mathcal{B}$, the LLL algorithm outputs another basis $\mathcal{V} = \{v_1, \ldots, v_n\}$ of $\mathcal{L}$ such that

$$\|v_1\| \leq 2^{n/2} \det(\mathcal{L})^{1/n}.$$

Further, the new basis can be computed in time polynomial in n and the bitlength of the b_i. $\diamond$

Using the LLL algorithm, Coppersmith presented methods for finding small integer solutions to bivariate polynomials [10] and for finding small solutions for univariate polynomials modulo some integer of unknown factorization [11]. These techniques have been extended to polynomials with more variables, but each extension is only a heuristic as they all depend on an assumption about algebraic independence (see Boneh and Durfee [6] for example). Even though the extensions cannot be proven, as of yet, they do work well in practice. We will make use of the following result concerning the solutions of linear multivariate polynomials (see Jochemsz and May [21] for more details).

Theorem 5: Let $f(x_1, \ldots, x_r)$ be a linear polynomial with integer coefficients. Let $X_1, \ldots, X_r$ be positive integers, $W = \|f(x_1 X_1, \ldots, x_r X_r)\|_2$ and N be a sufficiently large integer with unknown factorization. Given $(y_1, \ldots, y_r) \in \mathbb{Z}^r$ satisfying $|y_1| < X_1, \ldots, |y_r| < X_r$, if $(y_1, \ldots, y_r)$ is a root of f and $\prod_{i=1}^{r} X_i < W$, or if $(y_1, \ldots, y_r)$ is a root of f modulo N and $\prod_{i=1}^{r} X_i < N$ then for sufficiently large W or N, respectively, we can compute $(y_1, \ldots, y_r)$ in polynomial time, provided a common assumption about the algebraic independence of reduced vectors holds. $\diamond$

B. Security of Scheme I

In this section we consider the security of Dual RSA when the public key e is small and the private key d is large (likely the size of the moduli). We begin with an attack that recovers the parameters k_1 and k_2 from the key equations and then present two attacks that use these values to break the system.

1) Finding k_1 and k_2: We show that k_1 and k_2 can be recovered with a lattice-based method. Given the Dual RSA public information (e, N_1, N_2), notice that the difference between the key equations, $ed = 1 + k_1(N_1 - s_1)$ and $ed = 1 + k_2(N_2 - s_2)$, is given by

$$k_1(N_1 - s_1) = k_2(N_2 - s_2). \tag{7}$$

Let's assume that we know ℓ of the most significant bits of k_2 so that we can write $k_2 = K_2 + k_2'$, where k_2' is the only unknown and $|k_2'| < n_e + n_d - n - \ell$ (the value of K_2 will actually be determined by exhaustive search). Using this representation for k_2 in (7), and rearranging, we obtain

$$k_1 N_1 - k_2' N_2 + (K_2 s_2 + k_2' s_2 - k_1 s_1) = K_2 N_2 \tag{8}$$

which suggests that we look for small solutions of the polynomial

$$f(x, y, z) = N_1 x - N_2 y + z - K_2 N_2 \tag{9}$$

since $(x_0, y_0, z_0) = (k_1, k_2', K_2 s_2 + k_2' s_2 - k_1 s_1)$ is a root of $f(x, y, z)$. Defining the bounds $X = 2^{n_e + n_d - n}, Y = 2^{n_e + n_d - n - \ell}$ and $Z = 2^{n_e + n_d - n/2}$ we have that $|x_0| \leq X, |y_0| \leq Y$ and $|z_0| \leq Z$. From the integer case of Theorem 5, noting that $W = \|f(xX, yY, zZ)\|_\infty = 2^{n_e + n_d}$, we can recover (x_0, y_0, z_0) for sufficiently large N provided that $XYZ < W$, or $3n_e + 3n_d - 5n/2 - \ell < n_e + n_d$. Thus, to prevent k_1 and k_2 from being recovered from this lattice-based method (with an exhaustive search with parameter ℓ), the size of the public and private exponents should satisfy

$$n_e + n_d > 5n/4 + \ell/2. \tag{10}$$

Since the size of the private exponent in scheme I is, with very high probability, the same size as the moduli (i.e., $n_d = n$), this lattice based method is expected to work whenever $n_e < n/4 + \ell/2$, which is essentially the same as the bound obtained above using continued fractions with an exhaustive search for each convergent. Therefore, the size of the public exponent should satisfy

$$n_e > n/4 + \ell/2 \tag{11}$$

in order to prevent k_1 and k_2 from being recovered using this method. For 1024-bit moduli, allowing an exhaustive search with $\ell = 80$, the size of the private exponent should satisfy $n_e > 296$.

2) Small Public Exponent Attack With k_1 and k_2: The first attack using k_1 and k_2 exploits the special structure of the Dual RSA moduli. Recall from the key generation algorithm that $N_1 = p_1 q_1$ and $N_2 = p_2 q_2$, where $p_1 = x_1 x_2 + 1, q_1 = y_1 y_2 + 1, p_2 = x_1 y_2 + 1, q_2 = k_1 x_2 + 1$ and $y_1 = k_2$. This allows us to write $N_1 - 1$ and $N_2 - 1$ as the following system of equations with five unknowns:

$$\begin{aligned} N_1 - 1 &= x_1 x_2 k_2 y_2 + x_1 x_2 + k_2 y_2 \\ N_2 - 1 &= x_1 x_2 k_1 y_2 + k_1 x_2 + x_1 y_2 \end{aligned} \tag{12}$$

where $|x_1| = |k_1| = |k_2| = n_e$ and $|x_2| = |y_2| = n/2 - n_e$.

Now, let's assume that we are given k_1 and k_2. Notice that the system of equations for $N_1 - 1$ and $N_2 - 1$ in (12) is now reduced to only three variables: x_1, x_2, and y_2. If n_e is small enough, we can perform an exhaustive search for x_1 and solve the resulting system of two unknowns. Similarly, if $n/2 - n_e$ is small enough, we can perform a brute force search on x_2 (or y_2) and solve the resulting system of two unknowns. In either scenario, for each guess (of x_1, x_2 or y_2) we solve the resulting system for the remaining unknowns and test if the solution yields the desired factorization of N_1 and N_2.

To avoid this brute force attack the size of the public exponent should be chosen so that guessing x_1, x_2 or y_2 is infeasible. Thus, the size of the public exponent should satisfy

$$\ell \leq n_e \leq n/2 - \ell \tag{13}$$

where ℓ is chosen so that an exhaustive search of a space of size 2^ℓ is infeasible. For 1024-bit moduli and security parameter $\ell = 80$, the size of the public exponent should satisfy $80 \leq n_e \leq 432$.

3) Lattice-Based Attack With k_1 and k_2: Let's assume, again, that we are given k_1 and k_2. Using these we compute $k_1' = k_1/\gcd(k_1, k_2)$ and $k_2' = k_2/\gcd(k_1, k_2)$. Now, notice that dividing the difference between the key equations, $ed = 1 + k_1(N_1 - s_1)$ and $ed = 1 + k_2(N_2 - s_2)$, by $\gcd(k_1, k_2)$ yields

$$k_1'(N_1 - s_1) = k_2'(N_2 - s_2) \tag{14}$$

where s_1 and s_2 are the only unknown values. Since $\gcd(k_1', k_2') = 1$, we can reduce this equation modulo k_2' to obtain $s_1 \equiv N_1 \pmod{k_2'}$. Letting $\sigma_1 = N_1 \bmod k_2'$, we can write $s_1 = \sigma_1 + \tau_1 k_2'$, where τ_1 is the only unknown part (notice that $|\tau_1| = n/2 - n_e$). Substituting s_1 in (14) we obtain

$$k_1'(N_1 - \sigma_1 - \tau_1 k_2') = k_2'(N_2 - s_2) \tag{15}$$

which suggests that we look for small solutions, modulo N_1, of the polynomial

$$f_{N_1}(x, y) = k_1' k_2' x - k_2' y + k_2' N_2 + k_1' \sigma_1$$

since $(x_0, y_0) = (\tau_1, s_2)$ is a root of $f_{N_1}(x, y)$ modulo N_1. Defining the bounds $X = 2^{n/2 - n_e}$ and $Y = 2^{n/2}$ we then have $|x_0| \leq X$ and $|y_0| \leq Y$. From Theorem 5, we see that τ_1 and s_2 can be computed provided that N_1 is sufficiently large and $XY < N_1$ (or $n - n_e < n$), which is always satisfied. Therefore, for large enough N_1, we should always be able to recover τ_1 and s_2. Once τ_1 and s_2 are known we can easily compute $\varphi(N_1)$ and $\varphi(N_2)$ which then allow us to factor N_1 and N_2. Therefore, we conclude that Dual RSA is insecure whenever k_1' and k_2' are known. In order to ensure that k_1' and k_2' are not revealed from the method described above, the size of the public exponent should satisfy (11), i.e., $n_e > n/4 + \ell/2$ or $n_e > 296$ for 1024-bit moduli allowing an exhaustive search with $\ell = 80$.

4) Summary for Scheme I: To summarize, in order to avoid all the known attacks on Dual RSA-Small-e, allowing an exhaustive search with parameter ℓ, the size of the public exponent should satisfy

$$n_e + n_d > 5n/4 + \ell/2 \tag{16}$$

which simplifies to

$$n_e > n/4 + \ell/2 \tag{17}$$

when the private exponent is the same size as the moduli (which is the expected case for scheme I). Also, in the case that k_1 and k_2 might be known to the adversary, the public exponent should also satisfy

$$n_e < n/2 - \ell \tag{18}$$

to avoid the attack from Section V-B2.

C. Security of Scheme II

In this section we consider the security of Dual RSA when the private key d is small and public key e is large (likely the size of the moduli).

All of the known small private exponent attacks on RSA also apply to Dual RSA (scheme II). These include Wiener's continued fraction attack [44], Boneh and Durfee's lattice-based attack [6] and Blömer and May's lattice-based attack [4]. We summarize the strongest results, by Boneh and Durfee, here. From [6, Sec. 6], we see that, in general, RSA is considered unsafe when the size of the private exponent satisfies

$$n_d < \frac{7n}{6} - \frac{1}{3}\sqrt{n^2 + 6nn_e}. \tag{19}$$

When the public exponent is the same size as the modulus a stronger result is known, [6, Sec. 5], which shows that RSA is unsafe when the size the of private exponent satisfies

$$n_d < (1 - 1/\sqrt{2})n \approx 0.292n \tag{20}$$

TABLE III
EXPERIMENTAL RESULTS FOR LATTICE-BASED ATTACK ON SCHEME II

n_d	337	338	339	340	341	342
success rate (%)	100.0	98.5	83.6	17.4	0.2	0.0

In addition to these attacks, notice that each of the attacks on scheme I from Section V-B can also be mounted against scheme II. Thus, Dual RSA-Small-d should be considered insecure when the size of the private (and public) exponent satisfies

$$n_d + n_e < 5n/4 + \ell/2. \tag{21}$$

As the size of the public exponent decreases, this attack is stronger than the attack behind (19). Below, we present a new attack on scheme II that is stronger than all of these attacks.

1) Lattice-Based Attack: Given the Dual RSA public information (e, N_1, N_2), consider the following three equations:

$$\begin{array}{llll} Ad & & & = Ad \\ ed & -k_1 N_1 & & = 1 - k_1 s_1 \\ ed & & -k_2 N_2 & = 1 - k_2 s_2. \end{array} \tag{22}$$

This is simply the key equations along with the trivial equation $Ad = Ad$, where A is some integer. Letting

$$\mathcal{B} = \begin{bmatrix} A & e & e \\ 0 & -N_1 & 0 \\ 0 & 0 & -N_2 \end{bmatrix} \tag{23}$$

we can write the system of (22) as the vector-matrix equation

$$(d, k_1, k_2)\mathcal{B} = (Ad, 1 - k_1 s_1, 1 - k_2 s_2). \tag{24}$$

Let $v = (Ad, 1 - k_1 s_1, 1 - k_2 s_2)$ and let $A = eN^{-1/2}$ ($|A| = n_e - n/2$) so that each component of v is the same size and $\|v\|_2 < \sqrt{3} \cdot 2^{n_d + n_e - n/2}$. The hope is that v will be the smallest vector in the lattice $\mathcal{L}$ generated by the rows in $\mathcal{B}$. If this is true then we can easily compute the private exponent d by simply finding the smallest vector in $\mathcal{L}$ and dividing the first component by A.

Now, Theorem 3 tells us that a smallest vector in $\mathcal{L}$ is bound in size by $\sqrt{3}\det(\mathcal{L})^{1/3} = \sqrt{3} \cdot 2^{(n_e + 3n/2)/3}$, since $\det(\mathcal{L}) = AN_1N_2$. So, we see that a necessary condition for v to be a smallest vector in $\mathcal{L}$ is that

$$\sqrt{3} \cdot 2^{n_d + n_e - n/2} < \sqrt{3} \cdot 2^{(n_e + 3n/2)/3}$$

which simplifies to

$$n_d + 2n_e/3 < n. \tag{25}$$

When the public exponent is the same size as the moduli (which is expected), this condition further simplifies to

$$n_d < n/3. \tag{26}$$

Using the LLL algorithm with $\mathcal{B}$ as input, when the size of the private exponent satisfies (25) or (26), it is hoped that the smallest vector in the new reduced lattice basis will be v. This will happen if v is the smallest vector in $\mathcal{L}$ and if all other vectors in $\mathcal{L}$ are sufficiently larger than v. Since we cannot prove that this is the case, we rely on experiments to demonstrate the effectiveness of the attack. In Table III we show some experimental results of this attack using 1024-bit Dual RSA moduli and 1000 trials for each private exponent size. As can be seen, the attack is quite successful until roughly $n_d > 339$ and then quickly becomes very ineffective. Since $1024/3 = 341 + 1/3$, we see that the attack (for 1024-bit moduli) is practically successful up to almost the heuristic bound of $n/3$.

The key generation algorithm for scheme II computes the public key after choosing the private key. Since the public exponent is computed as the inverse of the private exponent modulo $\varphi(N_1)$, it is expected that the size of the public exponent will be roughly the same size as

the moduli. Because of this, it is difficult to generate instances of Dual RSA-Small-d with public exponents significantly smaller than N, and is the reason for only testing the attack on instances where the public exponent is roughly the same size as the moduli. Based on the success of the attack in this case, we believe that the attack will work when e is smaller than N.

2) Summary for Scheme II: In summary, in order to avoid all the known attacks on Dual RSA-Small-d, allowing an exhaustive search with parameter ℓ, the size of the public exponent should, in general, satisfy

$$n_d + n_e > 5n/4 + \ell/2 \tag{27}$$

$$n_d > \frac{7n}{6} - \frac{1}{3}\sqrt{n^2 + 6nn_e} \tag{28}$$

and

$$n_d > n/3 \tag{29}$$

when the public exponent is the same size as the moduli (which is the expected case for scheme II).

D. Security of Scheme III

For the security consideration of Dual Generalized Rebalanced-RSA we first review the known results for Generalized Rebalanced-RSA and then present a new attack.

From Sun *et al.* [36], (see [15] and [3] for a similar analysis) the public and private CRT-exponents should satisfy

$$\begin{aligned} &5n_d + 2n_e < 2n + \ell \\ &3n_d + 2n_e < 3n/2 + \ell_1 \quad \text{or} \quad n_e < n/4 - \ell_2 \\ &\quad \text{where } \ell = \ell_1 + \ell_2 \\ &6n_d + 3n_e > 5n/2 + \ell \\ &\quad n_d > 2\ell \end{aligned} \tag{30}$$

in order to avoid all known attack allowing at most 2^ℓ work.

1) Small n_k Attack: In this section we present an attack when the security parameter in the key generation algorithm is small (i.e., n_k is small). Recall from the key generation algorithm that

$$\begin{aligned} &N_1 = p_1 q_1 \quad N_2 = p_2 q_2 \\ &p_1 = p_a p_b + 1 \quad p_2 = p_{a'} p_b + 1 \\ &q_1 = q_a q_b + 1 \quad q_2 = q_{a'} q_b + 1 \\ &p_a = p_{a_1} p_{a_2} \cdots p_{a_{(k-1)}} p_{a_k} \quad p_{a'} = p_a k_{p1} / p_{a_{i'}} \\ &q_a = q_{a_1} q_{a_2} \cdots q_{a_{(k-1)}} q_{a_k} \quad q_{a'} = q_a k_{q_1} / q_{a_{j'}} \end{aligned} \tag{31}$$

for some i' and $j' \in \{1, \ldots, k-1\}$, where $k = \lceil (n/2 - n_e)/n_k \rceil$, $|p_{a_i}| = |k_{p_1}| = |k_{q_1}| = |q_{a_i}| = n_k$ for all $i \in \{1, \ldots, k-1\}$, and $|p_b| = |q_b| = n_e$. From this, notice that

$$\begin{aligned} \frac{N_1 - 1}{p_{a'} p_b q_{a'} q_b} &= \frac{p_a p_b q_a q_b + p_a p_b + q_a q_b}{p_{a'} p_b q_{a'} q_b} \\ &= \frac{p_{a_{i'}} q_{a_{j'}}}{k_{p_1} k_{q_1}} + \frac{p_a p_b + q_a q_b}{p_{a'} p_b q_{a'} q_b} \\ &< \frac{p_{a_{i'}} q_{a_{j'}}}{k_{p_1} k_{q_1}} + \frac{2 \cdot 2^{n/2}}{\left(\frac{1}{2} 2^{n/2}\right)^2} \end{aligned}$$

and

$$\begin{aligned} \frac{N_2 - 1}{p_{a'} p_b q_{a'} q_b} &= \frac{p_{a'} p_b q_{a'} q_b + p_{a'} p_b + q_{a'} q_b}{p_{a'} p_b q_{a'} q_b} \\ &= 1 + \frac{p_{a'} p_b + q_{a'} q_b}{p_{a'} p_b q_{a'} q_b} > 1. \end{aligned}$$

Dividing the first inequality by the second yields

$$\frac{N_1 - 1}{N_2 - 1} < \frac{p_{a_{i'}} q_{a_{j'}}}{k_{p_1} k_{q_1}} + \frac{8}{2^{n/2}} \tag{32}$$

which suggests that $(p_{a_{i'}} q_{a_{j'}})/(k_{p_1} k_{q_1})$ might be recovered by one of the convergents in the continued fraction expansion of $(N_1 - 1)/(N_2 - 1)$. In order to use the result of Theorem 2, we need

$$\left| \frac{N_1 - 1}{N_2 - 1} - \frac{p_{a_{i'}} q_{a_{j'}}}{k_{p_1} k_{q_1}} \right| < \frac{8}{2^{n/2}} < \frac{1}{2\,(k_{p_1} k_{q_1})^2}$$

which is satisfied whenever $k_{p1} k_{q1} < 2^{n/4-2}$ (or equivalently, when $n_k < n/8 - 1$). Thus, whenever $n_k < n/8 - 1$, one of the convergents in the continued fraction expansion of $(N_1 - 1)/(N_2 - 1)$ will be $p_{a_{i'}} q_{a_{j'}} / k_{p_1} k_{q_1}$ in lowest terms. As shown by Verheul and van Tilborg [42] and Dujella [13], we can include an exhaustive search on each convergent to increase the bound of the sufficient condition. For an exhaustive search of a search space of size 2^ℓ, the bound can be increased by $\ell/2$ (see [42] or [13] for details). Allowing such an exhaustive search for each convergent gives the new sufficient condition

$$n_k < n/8 - 1 + \ell/2. \tag{33}$$

Since $p_{a_{i'}} = k_{p2}$ and $q_{a_{j'}} = k_{q_2}$, if (33) is satisfied then one of the convergents will give k_1' and k_2' such that $k_2'/k_1' = k_{p_2} k_{q_2} / k_{p_1} k_{q_1}$ (i.e., k_1' and k_2' are exactly the same as in the previous attack). Therefore, we can apply the same lattice-based method from Section V-B3 to factor N_1 and N_2 provided that the moduli are sufficiently large. In order to avoid this attack, the security parameter n_k should satisfy

$$n_k > n/8 + \ell/2 - 1 \tag{34}$$

where we allow an exhaustive search of a space of size 2^ℓ for each convergent. Combining (34) with the condition that $n_e + n_d = n/2 + n_k$, we see that the size of the public and CRT exponents should satisfy

$$n_e + n_d > 5n/8 + \ell/2 - 1 \tag{35}$$

in order to avoid this attack.

2) Summary for Scheme III: In summary, to avoid all the known attacks on Dual Generalized Rebalanced-RSA, allowing an exhaustive search with parameter ℓ, the size of the key generation parameters n_e, n_d and n_k should satisfy all of

$$\begin{aligned} n_k &> n/8 + \ell/2 - 1 \\ n_e + n_d &> 5n/8 + \ell/2 - 1 \\ 5n_d + 2n_e &> 2n + \ell. \end{aligned} \tag{36}$$

In Fig. 1, we illustrate which key generation parameters are unsafe (area under the curves) for Dual Generalized Rebalanced-RSA with 1024-bit moduli. Also included are the attacks against Generalized Rebalanced-RSA.

E. Summary of Security

In Table IV, we give restrictions on the parameter selection (size of exponents) for each scheme. When the parameter restrictions are satisfied, the system is secure against all known attacks requiring no more than 2^ℓ work.

VI. Discussion

The motivation for Dual RSA is to reduce (key) storage requirements in systems that require two instances of RSA while retaining as much of the functionality as simply using two instances of RSA (e.g., using small public or private exponent, using small public and CRT-exponents).

We show the minimal memory requirements for three RSA variants, including Dual RSA, two instances of RSA ($2 \times$ RSA), and Twin RSA,

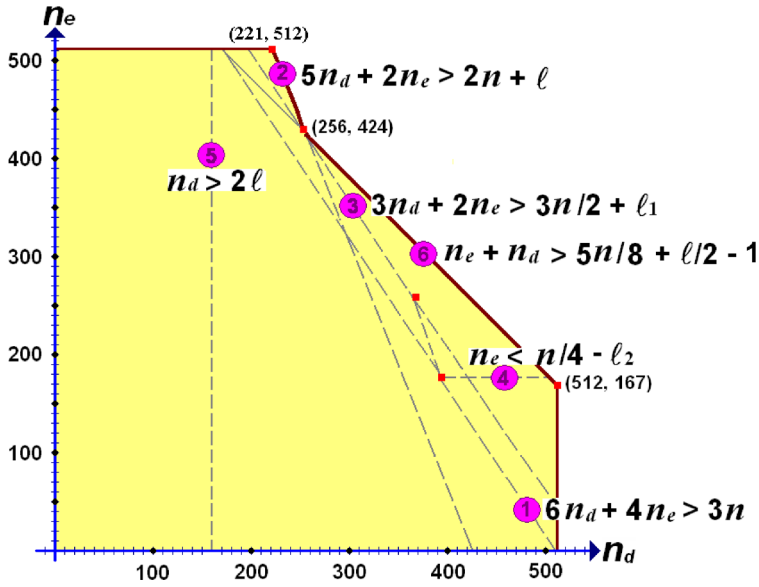

Fig. 1. Safe parameter choices for scheme III.

TABLE IV
SUMMARY OF SAFE DUAL RSA PARAMETERS

Scheme	Restrictions	Comments
I	$n_e > 5n/4 + \ell/2 - n_d$	$n_d \ll n$ (§V-B.1, V-B.3)
	$n_e > n/4 + \ell/2$	$n_d \approx n$ (§V-B.1, V-B.3)
	$n_e < n/2 - \ell$	known k_1 and k_2 (§V-B.2)
II	$n_d > 5n/4 - n_e + \ell/2$	$n_e \ll n$ (§V-C.1)
	$n_d > 7n/6 - \sqrt{n^2 + 6nn_e}/3$	$n_e \ll n$ (from [6])
	$n_d > n/3$	$n_e \approx n$ (§V-C.1)
III	$n_k > n/8 + \ell/2 - 1$	(§V-D.1)
	$n_e + n_d > 5n/8 + \ell/2 - 1$	(§V-D.1)
	$5n_d + 2n_e > 2n + \ell$	(see [36])

TABLE V
MEMORY REQUIREMENTS FOR SMALL EXPONENT RSA

Variant	Dual RSA		2×RSA	
Type	Small-e	Small-d	Small-e	Small-d
$\lvert N_1\rvert + \lvert N_2\rvert$	$2 \times n$	$2 \times n$	$2 \times n$	$2 \times n$
$\lvert e_1\rvert + \lvert e_2\rvert$	$n/4 + \ell/2$	n	17	$2 \times n$
$\lvert d_1\rvert + \lvert d_2\rvert$	n	$n/3$	$2 \times n$	$2 \times 0.29n$
Memory Requirement	$3.25n + \ell/2$	$3.333n$	$4n + 17$	$4.58n$

Variant	Twin RSA	
Type	Small-e	Small-d
$\lvert N_1\rvert + \lvert N_2\rvert$	$n + \lvert\Delta\rvert$	$n + \lvert\Delta\rvert$
$\lvert e_1\rvert + \lvert e_2\rvert$	17	$2 \times n$
$\lvert d_1\rvert + \lvert d_2\rvert$	$2 \times n$	$2 \times 0.29n$
Memory Requirement	$3n + 17 + \lvert\Delta\rvert$	$3.58n + \lvert\Delta\rvert$

TABLE VI
MEMORY REQUIREMENTS FOR REBALANCED-RSA TYPE

Variant	Dual RSA	2×RSA	Twin RSA
Type	Generalized Rebalanced-RSA	Rebalanced-RSA	Rebalanced-RSA
$\lvert N_1\rvert + \lvert N_2\rvert$	$2 \times n$	$2 \times n$	$n + \lvert\Delta\rvert$
$\lvert e_1\rvert + \lvert e_2\rvert$	$n/2$	$2 \times n$	$2 \times n$
$\lvert d_p\rvert + \lvert d_q\rvert$	$n/5 + \ell/5$	$2 \times 2\ell$	$2 \times 2\ell$
Memory Requirement	$2.7n + \ell/5$	$4n + 4\ell$	$3n + \lvert\Delta\rvert + 4\ell$

in Tables V and VI. For each variant we also consider three types, including Small-e, Small-d, and Rebalanced-RSA. The modulus form we consider here is the product of two prime-factors, i.e., $N = pq$. For the case of multipower (or multiprime) modulus ([18], [29], [40]), e.g., $N = p^2q$, Dual RSA still can be designed with multi-power (or multi-prime) moduli to achieve the purposes of saving memory requirement and efficient encryption and decryption. Also, Twin RSA [26] still can be designed with multi-prime moduli, but we do not compare their saving requirements here. Tables V and VI show the number of bits needed to store both public and private keys as a function of the bit length of the moduli N_1 and N_2, the difference between moduli Δ (for Twin RSA) and the security parameter ℓ (usually set $\ell = 80$). Notice that in the case of $2 \times$ RSA, we may choose the same public exponent e (usually set $e = 2^{16} + 1$) for Small-e type in order to save the memory requirement. The same way can be applied to Twin RSA with Small-e type. However, we do not suggest using the same private exponent in the case of Small-d type for $2 \times$ RSA and Twin RSA due to the security consideration. As can be seen in the security analysis of Dual RSA (see Table IV), using the same private exponents results in the requirement of higher security boundary against small exponent attacks. Up to now there is no discussion about whether using the same small private exponents in $2 \times$ RSA or Twin RSA is secure or not. Thus we use two different small private exponents in $2 \times$ RSA and Twin RSA with Small-d type. The same strategy is considered in Rebalanced-RSA type (see Table VI). Also, it is an interesting problem to look at security bounds when using the same small private exponents in $2 \times$ RSA or Twin RSA.

As can be seen in Table V, both Dual RSA and Twin RSA allow for substantial memory savings when ℓ and Δ are relatively small. In particular, Dual RSA requires the least amount of memory when using a small public or private exponents compared with $2 \times$ RSA. While compared with Twin RSA, Dual RSA requires less memory than that in Twin RSA with Small-d type, but more memory in Small-e type. In addition, Twin RSA may not be well suited for the threshold technique for authentication/secrecy in Section IV-B2. For a selected threshold h , all moduli (for all users) must have the same MSBs as h, that is approximately $(n - |\Delta|)$ bits. Consequently, the distribution of moduli will be abnormal because these moduli will be located in a narrow range $(h - \Delta, h + \Delta)$. This may lead to some powerful attacks if Δ is not large enough. A possible solution is to increase the size of Δ. However,

it is still unknown what size of the Δ is sufficient to make these RSA keys secure. Moreover making the Δ large also incurs more storage cost.

For Rebalanced-RSA type, as can be seen in Table VI, Dual RSA requires the least amount of memory. Depending on the parameters used there can be a substantial memory savings when using Dual RSA. For example, using $n_e \approx n/2$ in Dual RSA corresponds to a saving of about $1.3n$ bits for $2 \times$ RSA and about $0.3n$ bits for Twin RSA.

From the security analysis, Table IV, we see that there is a trade-off when using Dual RSA compared to using two instances of RSA (or Twin RSA). On the one hand, Dual RSA allows for reduced (key) storage requirements as shown above (i.e., reduced space complexity). On the other hand, the size of unsafe exponents is increased with Dual RSA, which increases encryption and/or decryption time (i.e., increased computational complexity). In addition, from Section III-D, the computational complexity of the key generation algorithms is also increased. Thus, the reduced space complexity of Dual RSA comes at a cost.

Therefore, in situations that require two instances of RSA (*cf.* Section IV) where reducing memory requirements is more important than the computational costs, in some constrained devices, for example, we view Dual RSA as a potentially good candidate. Of course, as with all new cryptosystems and new variants of trusted cryptosystems, the security of the system needs to be scrutinized more.

APPENDIX

We give an alternate key generation algorithm for Dual RSA-Small-e that outputs public/private key pairs with a specified public exponent. Let e be an n_e-bit positive integer. Given (e, n_e, n) as input the alternate key generation algorithm is as follows.

Algorithm KEY-GEN-SCHEME-I-ALT

Input: (e, n_e, n) such that $n_e < n/2$.

1) Randomly select an n_e-bit integer x_1 and an $(n/2 - n_e)$-bit integer x_2 such that $p_1 = x_1 x_2 + 1$ is prime and $\gcd(e, x_1 x_2) = 1$.
2) Randomly select an $(n/2 - n_e)$-bit integer y_2 such that $p_2 = x_1 y_2 + 1$ is prime and $\gcd(e, y_2) = 1$.
3) Randomly select an n_e-bit integer y_1 such that $q_1 = y_1 y_2 + 1$ is prime and $\gcd(e, y_1) = 1$.
4) Compute d and k_1 satisfying $ed = 1 + k_1(p_1 - 1)(q_1 - 1)$.
5) If $q_2 = k_1 x_2 + 1$ is not prime then go back to step 3.
6) Let $N_1 = p_1 q_1, N_2 = p_2 q_2$, and $k_2 = y_1$.
7) Output the public key (e, N_1, N_2) and the private key (d, p_1, q_1, p_2, q_2).

Output: (e, N_1, N_2) and (d, p_1, q_1, p_2, q_2). $\diamond$

The correctness of this algorithm follows from the same reasoning as the first algorithm (see (5)).

ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers for their valuable comments and suggestions on the part of security analysis, which certainly led to improvements of this correspondence.

REFERENCES

[1] S. Berkovits, "Factoring via superencryption," *Cryptologia*, vol. 6, no. 3, pp. 229–237, 1982.

[2] M. Bellare and P. Rogaway, "Optimal asymmetric encryption—How to encrypt with RSA," in *Advances in Cryptology—EUROCRYPT'94*, ser. Lecture Notes in Computer Science, A. D. Santis, Ed. New York: Springer, 1995, vol. 950, pp. 92–111.

[3] D. Bleichenbacher and A. May, "New attacks on RSA with small secret CRT-exponents," in *Public Key Cryptology—PKC 2006*, ser. Lecture Notes in Computer Science. New York: Springer, 2006, vol. 3958, pp. 1–13.

[4] J. Blömer and A. May, Eds., "Low Secret Exponent RSA Revisited," in *Cryptography and Lattices, Int. Conf., CaLC 2001*, J. H. Silverman, Ed., 2001, vol. 2146, pp. 4–19, Springer, Lecture Notes in Computer Science.

[5] D. Boneh, "Twenty years of attacks on the RSA cryptosystem," *Notices of the American Mathematical Society*, vol. 46, no. 2, pp. 203–213, 1999.

[6] D. Boneh and G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1339–1349, Jul. 2000.

[7] D. Boneh, G. Durfee, and Y. Frankel, "An attack on RSA given a small fraction of the private key bits," in *Advances in Cryptology—ASIACRYPT'98*, ser. Lecture Notes in Computer Science, K. Ohta and D. Pei, Eds. New York: Springer, 1998, vol. 1514, pp. 25–34.

[8] D. Boneh and H. Shacham, "Fast variants of RSA," *CryptoBytes*, vol. 5, no. 1, pp. 1–9, 2002.

[9] D. Chaum, "Untraceable electronic mail, return address and digital pseudonyms," *Commun. ACM*, vol. 24, no. 2, pp. 84–88, Feb. 1981.

[10] D. Coppersmith, "Finding a small root of a bivariate integer equation; Factoring with high bits known," in *Advances in Cryptology—EUROCRYPT'96*, ser. Lecture Notes in Computer Science, U. M. Maurer, Ed. New York: Springer, 1996, vol. 1070, pp. 178–189.

[11] D. Coppersmith, "Finding a small root of a univariate modular equation," in *Advances in Cryptology—EUROCRYPT'96*, ser. Lecture Notes in Computer Science, U. M. Maurer, Ed. New York: Springer, 1996, vol. 1070, pp. 155–165.

[12] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter, "Low-exponent RSA with related message," in *Advances in Cryptology—EUROCRYPT'96*, ser. Lecture Notes in Computer Science, U. M. Maurer, Ed. : Springer, 1996, vol. 1070, pp. 1–9.

[13] A. Dujella, "Continued fractions and RSA with small secret exponent," *Tatra Mt. Math. Publ.*, vol. 29, pp. 101–112, 2004.

[14] S. D. Galbraith, C. Heneghan, and J. F. McKee, "Tunable balancing of RSA," in *Proc. Inf. Security and Privacy, 10th Australasian Conf., ACISP 2005*, C. Boyd and J. M. G. Nieto, Eds., 2005, vol. 3574, pp. 280–292, Springer, Lecture Notes in Computer Science.

[15] S. D. Galbraith, C. Heneghan, and J. F. McKee, Tunable Balancing of RSA, Full Version of [4] [Online]. Available: http://www.isg.rhul.ac.uk/sdg/full-tunable-rsa.pdf

[16] M. Gysin and J. Seberry, "Generalised cycling attacks on RSA and strong RSA primes," in *Proc. Inf. Security and Privacy, 4th Australasian Conf., ACISP 1999*, 1999, vol. 1587, pp. 149–163, Springer, Lecture Notes in Computer Science.

[17] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, 4th ed. Cambridge, U.K.: Oxford Univ. Press, 1960.

[18] D. Hühnlein, M. J. Jacobson, S. Paulus, and T. Takagi, "A cryptosystem based on non-maximal imaginary quadratic orders with fast decryption," in *Advances in Cryptology—EUROCRYPT'98*, ser. Lecture Notes in Computer Science. New York: Springer, 1998, vol. 1403, pp. 294–307.

[19] M. J. Hinek, "Another look at small RSA exponents," in *Topics in Cryptology-CT-RSA 2006*, ser. Lecture Notes in Computer Science, D. Pointcheval, Ed. New York: Springer, 2006, vol. 3860, pp. 82–98.

[20] M. Joye, J.-J. Quisquater, and T. Takagi, "How to choose secret parameters for RSA and its extension to elliptic curves," *Designs, Codes Cryptogr.*, vol. 23, no. 3, pp. 297–316, 2001.

[21] E. Jochemsz and A. May, "A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants," in *Advances in Cryptology—ASIACRYPT'06*, ser. Lecture Notes in Computer Science. New York: Springer, 2006, vol. 4284, pp. 267–282.

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

[23] H. W. Lenstra, "Factoring integers with elliptic curves," *Ann. Math.*, vol. 126, pp. 649–673, 1987.

[24] A. K. Lenstra, "Generating RSA moduli with a predetermined portion," in *Advances in Cryptology—ASIACRYPT'98*, ser. Lecture Notes in Computer Science, K. Ohta and D. Pei, Eds. New York: Springer, 1998, vol. 1514, pp. 1–10.

[25] A. K. Lenstra, "Unbelievable security. Matching AES security using public key systems," in *Advances in Cryptology—ASIACRYPT'01*, ser. Lecture Notes in Computer Science, C. Boyd, Ed. New York: Springer, 2001, vol. 2248, pp. 67–86.

[26] A. K. Lenstra, B. M. M. de Weger, T. RSA, E. Dawson, and S. Vaudenay, *Progress in Cryptology—Mycrypt 2005*, ser. Lecture Notes in Computer Science. New York: Springer, 2005, vol. 3715, pp. 222–228.

[27] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC, 1996, Discrete Mathematics and Its Applications.
[28] I. Niven and H. S. Zuckerman, *An Introduction to the Theory of Numbers*. New York: Wiley, 1991.
[29] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *Advances in Cryptology—EUROCRYPT'98*, ser. Lecture Notes in Computer Science. New York: Springer, 1998, vol. 1403, pp. 308–318.
[30] J. M. Pollard, "Theorems on factorization and primality testing," *Proc. Cambridge Philosophical Soc.*, vol. 76, pp. 521–528, 1974.
[31] G. Qiao and K.-Y. Lam, "RSA signature algorithm for microcontroller implementation," in *Smart Card Research and Applications, CARDIS'98*, ser. Lecture Notes in Comput. Sci., J.-J. Quisquater and B. Schneier, Eds. New York: Springer, 1998, vol. 1820, pp. 353–356.
[32] J.-J. Quisquater and C. Couvreur, "Fast decipherment algorithm for RSA public key cryptosystem," *Electron. Lett.*, vol. 18, no. 21, pp. 905–907, Oct. 1982.
[33] R. Rivest, A. Shamir, and L. Aldeman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[34] R. Rivest and R. Silverman, Are "Strong Prime" Needed for RSA? Cryptology ePrint Archive Report 2001/007, 2001 [Online]. Available: http://eprint.iacr.org/2001/007
[35] R. D. Silverman, "Fast generation of random, strong RSA primes," *CryptoBytes*, vol. 3, no. 1, pp. 9–13, 1997.
[36] H.-M. Sun, M. J. Hinek, and M.-E. Wu, On the design of Rebalanced-RSA, revised version of [37] Centre for Applied Cryptographic Research, Technical Report CACR 2005-35, 2005 [Online]. Available: http://www.cacr.math.uwaterloo.ca/techreports/2005/cacr2005-35.pdf
[37] H.-M. Sun and M.-E. Wu, An approach towards Rebalanced RSA-CRT with short public exponent Cryptology ePrint Archive, Report 2005/053, 2005 [Online]. Available: http://eprint.iacr.org/2005/053
[38] H.-M. Sun and C.-T. Yang, "RSA with balanced short exponents and its application to entity authentication," in *Public Key Cryptology—PKC 2005, Lecture Notes in Computer Science*. New York: Springer, 2005, vol. 3386, pp. 199–215.
[39] H.-M. Sun, W.-C. Yang, and C.-S. Laih, "On the design of RSA with short secret exponent," in *Advances in Cryptology—ASIACRYPT'99*, ser. Lecture Notes in Computer Science, K.-Y. Lam, E. Okamoto, and C. Xing, Eds. Berlin: Springer, 1999, vol. 1716, pp. 150–164.
[40] T. Takagi, "Fast RSA-type cryptosystem modulo p^2q," in *Advances in Cryptology-CRYPTO'98*, ser. Lecture Notes in Computer Science. New York: Springer, 1998, vol. 1462, pp. 318–326.
[41] S. A. Vanstone and R. J. Zuccherato, "Short RSA keys and their generation," *J. Cryptol.*, vol. 8, no. 2, pp. 101–114, Mar. 1995.
[42] E. R. Verheul and H. C. A. van Tilborg, "Cryptanalysis of 'less short' RSA secret exponents," *Appl. Algebra Eng. Commun. Comput.*, vol. 8, no. 5, pp. 425–435, 1997.
[43] G. Wang, Bibliography on Blind Signatures [Online]. Available: http://www.i2r.a-star.edu.sg/icsd/staff/guilin/bible/blind-sign.htm [ONLINE], Available
[44] M. J. Wiener, "Cryptanalysis of short RSA secret exponents," *IEEE Trans. Inf. Theory*, vol. 36, no. 3, pp. 553–559, May 1990.

Applications of LDPC Codes to the Wiretap Channel

Andrew Thangaraj, *Member, IEEE*,
Souvik Dihidar, A. R. Calderbank, *Fellow, IEEE*,
Steven W. McLaughlin, *Fellow, IEEE*, and Jean-Marc Merolla

***Abstract*—With the advent of quantum key distribution (QKD) systems, perfect (i.e., information-theoretic) security can now be achieved for distribution of a cryptographic key. QKD systems and similar protocols use classical error-correcting codes for both error correction (for the honest parties to correct errors) and privacy amplification (to make an eavesdropper fully ignorant). From a coding perspective, a good model that corresponds to such a setting is the wire tap channel introduced by Wyner in 1975. In this correspondence, we study fundamental limits and coding methods for wire tap channels. We provide an alternative view of the proof for secrecy capacity of wire tap channels and show how capacity achieving codes can be used to achieve the secrecy capacity for any wiretap channel. We also consider binary erasure channel and binary symmetric channel special cases for the wiretap channel and propose specific practical codes. In some cases our designs achieve the secrecy capacity and in others the codes provide security at rates below secrecy capacity. For the special case of a noiseless main channel and binary erasure channel, we consider encoder and decoder design for codes achieving secrecy on the wiretap channel; we show that it is possible to construct linear-time decodable secrecy codes based on low-density parity-check (LDPC) codes that achieve secrecy.**

***Index Terms*—Binary erasure channels, low-density parity-check (LDPC) codes, secrecy capacity, secrecy codes, wire tap channels.**

I. Introduction and Motivation

The notion of communication with perfect security was defined in information-theoretic terms by Shannon [1]. Suppose a k-bit message $\mathbf{M}$ is to be transmitted securely from Alice to Bob across a public channel. Perfect security is said to be achieved if the encoding of $\mathbf{M}$ into a transmitted word $\mathbf{X}$ is such that the mutual information $I(\mathbf{M};\mathbf{X}) = 0$. From this definition, Shannon concluded that Alice and Bob should necessarily share k bits of key for achieving perfect security.

An alternative notion of communication with security was introduced by Wyner [2]. Wyner introduced the wire tap channel, which has matured into a system depicted in Fig. 1. In a wire tap channel, the honest parties Alice and Bob are separated by a channel C1 called the main channel. The important modification when compared to Shannon's study of security is that any eavesdropper Eve observes information transmitted by Alice through another channel C2 called

Manuscript received December 22, 2005; revised October 19, 2006. This work was performed while A. Thangaraj, S. Dihidar, S. McLaughlin, and J.-M. Merolla were with the GTL-CNRS Telecom Lab, Metz, France. The material in this correspondence was presented in part at the IEEE International Symposium on Information Theory, Adelaide, Australia, September 2005.

A. Thangaraj is with the Department of Electrical Engineering, Indian Institute of Technology Madras, Chennai 600036, India (e-mail: andrew@iitm.ac.in).

S. Dihidar is with Marvell Semiconductor, Santa Clara, CA 95054 USA He is also with the Electrical and Computer Engineering Department, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: dihidar@ece.gatech.edu).

A. R. Calderbank is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: calderbk@math.princeton.edu).

S. W. McLaughlin is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA . He is also with GT-CNRS Unité Mixte Internationale (UMI 2958), Metz, France (e-mail: swm@ece.gatech.edu).

J.-M. Merolla is with the CNRS FEMTO Lab, Besançon, France (e-mail: merolla@georgiatech-metz.fr).

Communicated by M. P. C. Fossorier, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2007.901143