On the Security of RSA with Primes Sharing Least-Significant Bits[*]

Ron Steinfeld
Dept. of Computing,
Macquarie University, Australia
rons@ics.mq.edu.au

Yuliang Zheng
Dept. of Software and Information Systems,
University of North Carolina at Charlotte, USA
yzheng@uncc.edu

Abstract

We investigate the security of a variant of the RSA public-key cryptosystem called *LSBS-RSA*, in which the modulus primes share a large number of least-significant bits. We show that low public-exponent LSBS-RSA is inherently resistant to *Partial Key Exposure* (PKE) attacks in which least-significant bits of the secret exponent are revealed to the attacker, and in particular that the Boneh-Durfee-Frankel PKE attack [5] on low public-exponent RSA is less effective for LSBS-RSA systems than for standard RSA. On the other hand, we show that large public-exponent LSBS-RSA is more vulnerable to such attacks than standard RSA. An application to server-aided RSA signature generation is proposed.

Key Words. RSA cryptosystem, Communication security, Cryptanalysis, Partial key exposure, Boneh-Durfee-Frankel attack, Coppersmith algorithm, Least-Significant bits, Server-aided signature generation.

1 Introduction

The RSA public-key primitive has enjoyed very wide applicability in cryptography since its initial publication in 1978 [19]. For this reason, the security of RSA has been extensively analyzed under various attack scenarios. A research area of particular interest is the investigation of certain special variants of RSA which are attractive for increasing computational efficiency. Such studies are especially important due to the relatively low computational power of modern portable devices which need to perform cryptographic operations, such as smart cards and mobile phones.

In this paper, we study security properties of the RSA primitive under the following conditions:

(1) The prime factors p and q of the public RSA modulus $N = pq$ have exactly α equal least-significant bits, that is, $p - q = r \cdot 2^{\alpha}$ for some odd integer r. We call an RSA modulus N with this property an α-Least-Significant-Bit-Symmetric modulus, or α-LSBS for short.

(2) The β least-significant bits of the RSA secret exponent d are available to the attacker (e.g. they are included as part of the public-key). For this reason, this setting is called a *Partial Key Exposure* (PKE) attack scenario.

(3) The RSA public-exponent e has bit length γ.

[*]This is an extended version of an earlier paper presented at the Cryptographer's Track RSA Conference (CT-RSA 2001), April 8-12 2001, San Francisco, USA [20].

We refer to the above RSA system as (α, β, γ)-LSBS RSA. We will sometimes refer to the standard RSA system obtained when no bits of the secret exponent d are revealed, namely $(\alpha, 0, \gamma)$-LSBS RSA, as simply α-LSBS RSA. When the length γ of the public exponent e is a small constant, we call the system a *Low Public Exponent* system. When γ is a constant fraction of the modulus length n, we call the system a *Large Public Exponent* system.

We investigate the security of (α, β, γ)-LSBS RSA as a function of the system parameters (α, β, γ). As is usually the case in cryptography, our goal is two-way: (1) To understand which regions of the parameter space are insecure and must be avoided, and (2) To identify regions which appear secure and can lead to improved efficiency of RSA implementations. In the latter case, we call the cryptographic assumption that (α, β, γ)-LSBS RSA is a trapdoor one-way function, *the* (α, β, γ)-*LSBS RSA assumption.*

Our main results can be summarised as follows.

Analysis of Intractability of Boneh-Durfee-Frankel PKE Attack for Low Public Exponents. We analyze a PKE attack due to Boneh, Durfee and Frankel [5] for low public exponents in the general case of (α, β, γ)-LSBS RSA (Boneh, Durfee and Frankel analyzed only the case of $(1, \beta, \gamma)$-LSBS RSA). We show that the attack runs efficiently only when $\beta \geq n/4 + \alpha$. In particular, when $\beta = n/4$, the running-time of the attack is exponential in the number of shared Least-Significant (LS) bits α, and hence intractable for even moderately large α. The attack gives a *lower bound* on the insecurity of (α, β, γ)-LSBS RSA.

Reduction Between (α, β, γ)-LSBS and α-LSBS RSA for Low Public Exponents. We give a reduction which complements the above attack and gives an *upper bound* on the insecurity of (α, β, γ)-LSBS RSA (one wayness with β LS bits of d revealed) in terms of the insecurity of α-LSBS RSA (one-wayness with no LS bits of d revealed). The reduction shows that, for low public exponents, if $\beta \leq 2\alpha$ then the one-wayness of (α, β, γ)-LSBS RSA follows from the one-wayness of α-LSBS RSA.

PKE Attacks for Large Public Exponents. We show that, unlike the low-exponent case, for large public exponents it is very dangerous to reveal LS bits of d in an α-LSBS system with large α. Our best attack (for prime public exponents) runs efficiently when $\alpha \geq n/12$ and $\beta \geq \gamma = n/6$. As α increases, the required size of β and γ decreases; in particular, for $\alpha = (1 - \epsilon)n/4$ and any constant $\epsilon > 0$, our attack works efficiently already for $\beta \geq \gamma = \epsilon \cdot n/4$ (we remark for comparison that the best known attack on standard RSA moduli with β known LS bits of d and large public exponents works efficiently only for $\beta \geq n/2$ [3]).

Application to Server-Aided Signatures. We motivate the study of the security of LSBS-RSA by giving a practical application - namely fast server-aided RSA signature generation.

1.1 A Motivating Application

Let $N = pq$ denote an RSA modulus of length n bits, with p and q primes each of length about $n/2$ bits. It is common in practice to use a *low public exponent* in the RSA public key system [19]. For these variants the public exponent e is chosen to be a small value (e.g. 3), independent of the modulus length n. The advantage of this choice is that the encryption exponentiation $x \mapsto x^e \bmod N$, whose computational cost is linear in the length of e, can be performed very quickly. However, the length of the secret exponent $d = e^{-1} \bmod \phi(N)$ is still long, that is n bits, and hence the decryption exponentiation $x \mapsto x^d \bmod N$ remains a computational bottleneck when it is performed in a low speed device such as a smart card.

In many such cases, a possible solution for speeding up the decryption operation is to find a way for the low-speed device (which we hereafter refer to as the *card*) to use a powerful but publicly observable external *server* to perform some of the decryption computation, without leaking any useful secret knowledge (such as the prime factors of N) to the server. Such a scheme has been called a 'Server-Aided-Secret-Computation' (SASC), with the first such schemes for RSA proposed by Matsumoto, Kato and Imai [14]. Many such schemes have been proposed, but many have been shown to be insecure (see [17] for a recent example).

In Section 5 we show how our results on the security of α-LSBS can be used to construct a simple Server-Aided-Signature-Generation protocol. For a modulus length of $n = 2048$ bit the protocol reduces the computational cost for the card by a factor of up to about 10 with respect to the no-aid situation, while still achieving a provable security against passive attacks assuming only the one-wayness of α-LSBS RSA.

1.2 Related Work

A good general survey of attacks on the RSA system was given by Boneh [4].

In Asiacrypt '98, Boneh, Durfee and Frankel (BDF) described several interesting *Partial Key Exposure* (PKE) attacks on the RSA system [5] (see [6] for a revised version). As introduced above, in a PKE attack the attacker has access to a portion of the secret key, as well as the public key. In particular, for low exponent RSA, BDF showed how to factor N (and hence recover the whole secret exponent d) in time polynomial in n, given only the $n/4$ Least Significant (LS) bits of d. Our work shows that this result actually only holds in the case of α-LSBS moduli with small α, and that in general one in fact needs at least $n/4 + \alpha$ LS bits of d for the attack to be polynomial-time. Additional PKE attacks on RSA with large public exponent were found by Blömer and May [3].

Hinek, Low and Teske [10] investigate the security of *multi-prime* RSA, in which the modulus N has at least three prime factors. They conclude experimentally, that unlike our results for two prime factors, the running time of the BDF partial key exposure attack is not exponentially related to the number of shared LS bits of the prime factors (although they show that the attack is intractable in the case of multi-prime moduli due to other reasons).

The prime factors of the α-LSBS moduli studied in this paper have α equal *Least*-Signficant bits. One could also study the complementary case of RSA moduli with primes sharing *Most*-Significant (MS) bits, which means that p and q are 'close'. This case is of some historical significance, since there exists a simple factoring algorithm due to Fermat (see [18] or [8]) which factors $N = pq$ efficiently if $|p - q| \leq 2^{n/4}$. Our simple factoring algorithm for α-LSBS moduli when $\alpha \geq n/4$ (see Corollary 3.1) can be considered an analogue of Fermat's algorithm. DeWeger [8] shows that RSA systems in which p and q share MS bits are more vulnerable to attacks than random moduli when the *secret* exponent d is small.

Finally, we mention that Lenstra [11] discusses techniques for generating RSA moduli with portions of the *modulus* bits fixed to a desired value. These techniques also allow computational savings in certain cases (e.g. by using moduli which are close to a power of 2). However, unlike the moduli discussed by Lenstra, our proposed α-LSBS moduli have a potential speedup advantage by leaking bits of d.

1.3 Contents of the Paper

This paper is organized as follows. In Section 2, we review the RSA trapdoor one-way function, and present results on lattice algorithms and square-root algorithms modulo a power of 2, which will be used in the paper. Section 3 looks at the difficulty of factoring α-LSBS moduli and gives some important properties of these moduli. In Section 4 we investigate the security of α-LSBS RSA under Partial Key Exposure (PKE) attacks, for both small and large public exponents. In Section 5 we propose an application of α-LSBS RSA to fast server-aided RSA signature generation. Finally, Section 6 concludes with an open problem.

2 Preliminaries

2.1 The RSA Trapdoor One-Way Function

The well-known RSA trapdoor one-way function $\mathsf{RSA} = (\mathsf{GK}, \mathsf{f}, \mathsf{f}^{-1})$, introduced in [19], is defined as follows. On input a security parameter n, the key generation algorithm GK outputs a secret/public key pair (sk, pk).

The public key $pk = (N, e)$ consists of an RSA modulus $N = pq$ with p and q primes and $\lceil \log_2 N \rceil = n$ (i.e. N is an n-bit modulus). In this paper we will be interested in the case when N is an α-LSBS modulus, meaning that $p - q = r \cdot 2^\alpha$ for some odd r. We will also assume throughout the paper that N is *balanced*, meaning that that the primes p and q are of approximately the same length of $n/2$ bits. More precisely, we will assume that

$$p < q < 2N^{1/2} \leq 2 \cdot 2^{n/2}. \tag{1}$$

The public exponent e is an integer satisfying $\gcd(e, \phi) = 1$, where

$$\phi = (p-1)(q-1) = N + 1 - (p+q), \tag{2}$$

is Euler's phi function evaluated at N. We will use $\gamma = \lceil \log_2 e \rceil$ to denote the length of e. The trapdoor key is the secret exponent d defined as the multiplicative inverse e^{-1} of the public exponent e modulo ϕ. That is,

$$d = e^{-1} \bmod \phi. \tag{3}$$

Note that this key generation equation can also be written over the integers as

$$ed - 1 = k \cdot \phi, \tag{4}$$

for some quotient integer k. Note that from (3) we have that $d < \phi$ and hence $k = e \cdot (d/\phi) - 1/\phi < e$.

Given the public key (N, e) and input $x \in \mathbb{Z}_N^*$, the RSA trapdoor function evaluation algorithm f outputs $y = \mathsf{f}((N, e), x) = x^e \bmod N$ (this is also called the *encryption* operation). Given the secret trapdoor d, and an input $y \in \mathbb{Z}_N^*$ the RSA function inversion algorithm f^{-1} outputs $x = \mathsf{f}^{-1}((N, e, d), y) = y^d \bmod N$ (this is also called the *decryption* operation). The RSA trapdoor function is conjectured to be *one-way*. This means that, if p and q are chosen at random from an appropriate probability distribution, and x is chosen uniformly at random from $\mathbb{Z}_N^*$, then given (N, e, y) with $y = x^e \bmod N$, it is infeasible to compute x.

2.2 Lattice-Based Factoring Algorithm for Partially Known Factors

In this paper we make use of an important algorithm due to Coppersmith [7] for factoring integers with partially known prime factors. This algorithm was obtained by Coppersmith as an application of a general algorithm (based on lattice reduction) for finding small integer solutions to bivariate polynomials over the integers.

The following is a statement of this result for the general case when the least-significant bits are given as a remainder modulo an arbitrary integer r (the original result of Coppersmith was for bits modulo a power of 2— the generalized version given here was first stated in [5]).

Theorem 2.1. *Let $N = pq$ be an n-bit RSA modulus. If $r \geq 2^{n/4}$ and $p_0 = p \bmod r$ are given, then we can factor N in time polynomial in n.*

2.3 Square-Roots Modulo a Power of 2

In our analysis in Section 4.1.1, we need some elementary results on square-roots modulo a power of 2, and the computational complexity of finding these roots. These results are summarised in the following lemma.

Lemma 2.1. *(1) Let* $d \equiv 1 \pmod 8$ *and* $\gamma \geq 3$. *Then there are exactly four solutions in* $\mathbb{Z}_{2^\gamma}$ *to the congruence* $x^2 \equiv d \pmod{2^\gamma}$. *These solutions are of the form* $x = \pm s + \delta \cdot 2^{\gamma-1}$ *with* $\delta \in \{0,1\}$ *and* s *is any solution to* $x^2 \equiv d \pmod{2^\gamma}$. *Furthermore, there exists an algorithm that, given* d *and* γ, *computes these four solutions in time* $O(\gamma^2)$ *bit operations.*

(2) The set of solutions in $\mathbb{Z}_{2^\gamma}$ *to the modular equation* $x^2 \equiv c \pmod{2^\gamma}$ *is summarised as follows. Let* $m = m_2(c)$ *denote the multiplicity of 2 in* c *and let* $d = \mathrm{odd}(c) = c/2^{m_2(c)}$.

(2.1) If $\gamma \leq m$, *there are* $2^{\lfloor \gamma/2 \rfloor}$ *solutions* $x \equiv 0 \bmod 2^{\lceil \gamma/2 \rceil}$.

(2.2) If $\gamma > m$, *there are no solutions if* m *is odd. Otherwise, if* m *is even, there are three subcases.*

If $\gamma = m+1$, *there are* $2^{m/2}$ *solutions* $x \equiv 2^{m/2} \pmod{2^{m/2+1}}$.

If $\gamma = m+2$, *there are* $2 \cdot 2^{m/2}$ *solutions* $x \equiv \pm 2^{m/2} \pmod{2^{m/2+2}}$ *if* $d \equiv 1 \pmod 4$ *and none otherwise.*

If $\gamma \geq m+3$, *there are* $4 \cdot 2^{m/2}$ *solutions of the form* $x \equiv 2^{m/2}(\pm s + \delta \cdot 2^{\gamma-m-1}) \pmod{2^{\gamma-m/2}}$ *with* $\delta \in \{0,1\}$ *if* $d \equiv 1 \pmod 8$, *and no solutions if* $d \not\equiv 1 \pmod 8$. *Here* s *is any solution to* $s^2 \equiv d \pmod{2^{\gamma-m}}$.

Proof. Proof of (1). Well known - see, e.g. Exercise 7.9.38 in [1].

Proof of (2.1). Write $x = 2^\alpha r$ for r odd. Since $m \geq \gamma$, we have $x^2 \equiv 0 \pmod{2^\gamma}$, or equivalently $r^2 2^{2\alpha} \equiv 0 \pmod{2^\gamma}$. Since r is odd, this is equivalent to $2^{2\alpha} \equiv 0 \pmod{2^\gamma}$, or $\alpha \geq \lceil \gamma/2 \rceil$, meaning the solutions are $x \equiv 0 \pmod{2^{\lceil \gamma/2 \rceil}}$ and there are $2^{\gamma - \lceil \gamma/2 \rceil} = 2^{\lfloor \gamma/2 \rfloor}$ such solutions in $\mathbb{Z}_{2^\gamma}$.

Proof of (2.2). Since $m < \gamma$, we have from $x^2 \equiv c \equiv 2^m d \pmod{2^\gamma}$ that $x^2 \equiv 0 \pmod{2^m}$. Dividing by 2^m, we get $\frac{x^2}{2^m} \equiv d \pmod{2^{\gamma-m}}$. Writing $x = 2^\alpha r$ for r odd, we have $2^{2\alpha - m} r^2 \equiv d \pmod{2^{\gamma-m}}$. But since d is odd then so is $2^{2\alpha-m} r^2$, meaning $2\alpha - m = 0$, m must be even if a solution exists, $\alpha = m/2$, and

$$r^2 \equiv d \pmod{2^{\gamma-m}}. \tag{5}$$

If $\gamma = m+1$, then (5) becomes $r^2 \equiv 1 \pmod 2$, which has as a unique solution $r \equiv 1 \pmod 2$. Thus the solutions in this case are $x \equiv 2^{m/2} \pmod{2^{m/2+1}}$, as required.

If $\gamma = m+2$, then (5) becomes $r^2 \equiv 1 \pmod 4$, which has two solutions $r \equiv \pm 1 \pmod 4$ if $d \equiv 1 \pmod 4$ and none otherwise. Thus the solutions in this case are $x \equiv \pm 2^{m/2} \pmod{2^{m/2+2}}$, as required.

If $\gamma \geq m+3$, then by part (1) of the Lemma we have that if $d \equiv 1 \pmod 8$ then (5) has the four solutions $r \equiv \pm s + \delta \cdot 2^{\gamma-m-1} \pmod{2^{\gamma-m}}$ in $\mathbb{Z}_{2^{\gamma-m}}$, where $\delta \in \{0,1\}$ and s is any solution to (5). This gives $x \equiv 2^{m/2}(\pm s + \delta \cdot 2^{\gamma-m-1}) \pmod{2^{\gamma-m/2}}$, as required. If $d \not\equiv 1 \pmod 8$ then (5) has no solution for r because any such solution must satisfy $r^2 \equiv d \pmod 8$ and it is easily verified that the latter congruence has no solutions if $d \not\equiv 1 \pmod 8$ and d is odd. □

□

3 Factoring α-LSBS RSA Moduli

A fundamental question in investigating the security of α-LSBS RSA is clearly the computational complexity of factoring α-LSBS RSA moduli, since the difficulty of factoring is a necessary condition for the one-wayness of α-LSBS RSA. Little is currently known in this area, but in this section we summarize the current state of the art, as far as is known to the authors. We also show some important properties of α-LSBS moduli which will be used in later sections.

3.1 Relation to the Standard RSA Factoring Problem

It is clear that as the number of shared LS bits α of p and q are reduced, the problem of factoring an α-LSBS modulus approaches a standard problem of factoring a random RSA modulus. Indeed, when the prime factors p and q of N are chosen randomly and independently, one would heuristically expect that the probability that $N = pq$ is an α-LSBS modulus is about $1/2^\alpha$ for $\alpha \geq 1$. Thus a random RSA modulus is heuristically 'on average' a 2-LSBS RSA modulus. However, we will be interested in the case when α is much larger, meaning a constant fraction of the modulus length n. In this case, the probability that a random modulus is an α-LSBS modulus is exponentially small and so the problem of factoring α-LSBS moduli may become easier than the standard problem.

3.2 Relation to the One-Wayness of α-LSBS RSA

The only known attack on the one-wayness of α-LSBS RSA is to factor the α-LSBS modulus N. This situation is the same as for the standard RSA system, when no extra information is available [4]. Thus we conjecture that the breaking the one-wayness of α-LSBS RSA is as hard as the problem of factoring α-LSBS moduli.

3.3 Leakage of α LS bits of p and 2α LS bits of $p+q$

With current state of knowledge, the asymptotic complexity of factoring α-LSBS RSA moduli appears to remain about the same as that of the standard factoring problem until α reaches a threshold of $(1/4 - \epsilon)n$ for an arbitrarily small constant $\epsilon > 0$. The reason for the latter upper bound is that α-LSBS moduli 'leak' the α shared LS bits of p and q and also the 2α LS bits of $p + q$, as shown in the following lemma.

Lemma 3.1. *Let $N = pq$ denote an n-bit α-LSBS RSA modulus. There exists an algorithm A which, given N, computes in time $O(n^2)$: (1) Four candidates for the α shared LS bits of p and q, and (2) Four candidates for the 2α LS bits of $s = p + q$.*

Proof. The algorithm A works as follows. Writing $p = l + p_H \cdot 2^\alpha$ and $q = l + q_H \cdot 2^\alpha$ with $l < 2^\alpha$ representing the α shared LS bits of p and q, we see that $N \equiv pq \equiv l^2 \pmod{2^\alpha}$. Thus l is a solution to the modular quadratic

$$x^2 \equiv N \pmod{2^\alpha}. \tag{6}$$

So the algorithm A applies Lemma 2.1 to compute in time $O(n^2)$ at most 4 candidates for l. This proves part (1) of the lemma. To prove part (2), note that

$$N = pq = (l + p_H \cdot 2^\alpha) \cdot (l + q_H \cdot 2^\alpha) = l^2 + l(p_H + q_H)2^\alpha + p_H q_H 2^{2\alpha},$$

and

$$s = p + q = 2l + (p_H + q_H)2^\alpha.$$

Consequently $N - l^2 \equiv l(p_H + q_H)2^{\alpha} \pmod{2^{2\alpha}}$, and since l is odd, it has a multiplicative inverse l^{-1} modulo $2^{2\alpha}$ which can be computed in time $O(n^2)$. So A computes $s_H \overset{\text{def}}{=} l^{-1} \cdot (N - l^2) \bmod 2^{2\alpha}$ and then $s_0 \overset{\text{def}}{=} 2l + s_H \bmod 2^{2\alpha}$. Since $s_H \equiv (p_H + q_H)2^{\alpha} \pmod{2^{2\alpha}}$, we see that

$$s_0 \equiv 2l + (p_H + q_H)2^{\alpha} \equiv p + q \pmod{2^{2\alpha}},$$

as required (we get four candidates for s_0 from the four candidates for l). This completes the proof of part (2). $\square$

$\square$

From this lemma we immediately obtain the following corollary.

Corollary 3.1. *Let $N = pq$ denote an n-bit α-LSBS RSA modulus. If $\alpha \geq n/4$, then we can factor N in time polynomial in n.*

Proof. The factoring algorithm F applies Lemma 3.1 to compute four candidates for $s_0 \overset{\text{def}}{=} p + q \bmod 2^{2\alpha}$. But since $2\alpha \geq n/2$, we have from $p + q \leq 4\sqrt{N} \leq 2^{n/2+2}$ that s_0 provides all except possibly the 2 MS bits of $p + q$. Searching over these 2 bits gives 16 candidates for $s \overset{\text{def}}{=} p + q$. Since $p^2 - (p+q)p + pq = 0$, F knows that p is a solution to the quadratic

$$x^2 - s \cdot x + N = 0,$$

over the integers, which it can easily solve in polynomial time for each of the 16 candidates for s, checking each solution until a factor of N is found. $\square$

$\square$

Note that if α is smaller than $n/4$ by even an arbitrarily small constant fraction $\epsilon > 0$ of n, that is if $\alpha = (1-\epsilon)n/4$, the running time of the above factoring algorithm already becomes asymptotically exponential in n. More precisely, one has to try $2^{\epsilon \cdot n/2}$ values for s because there are $\epsilon \cdot n/2$ missing MS bits of s.

Note also that the above algorithm makes use of the $(1-\epsilon)n/2$ LS bits of s from part (2) of Lemma 3.1. We can instead use the $(1-\epsilon)n/4$ LS bits of p from part (1) of Lemma 3.1 and search the missing $\epsilon \cdot n/4$ MS bits of p to obtain the $n/4$ LS bits of p, using Coppersmith's polynomial-time factoring algorithm of Theorem 2.1 to try to factor N for each candidate. Although this algorithm remains exponential in n, the exponent of 2 is reduced by half to $\epsilon \cdot n/4$.

In conclusion, it appears at present that as long as ϵ is sufficiently large that a set of size $2^{\epsilon \cdot n/4}$ is infeasible to search, and n is sufficiently large that general-purpose factoring algorithms are infeasible, there is no efficient factoring algorithm for $(1-\epsilon)n/4$-LSBS RSA moduli. That is, we might conjecture that $T_{FL}(n, \epsilon) = \min(T_{CL}(n, \epsilon), T_{GF}(n))$, where $T_{FL}(n, \epsilon)$ is the time needed to factor an $(1-\epsilon)n/4$-LSBS RSA modulus, $T_{CL}(n, \epsilon) = O(2^{n/4 \cdot \epsilon})$ is the run-time of the factoring algorithm described above, and $T_{GF}(n)$ is the run-time of the best general-purpose factoring algorithm. As a typical example, it is estimated [12] that factoring a standard RSA modulus of length $n = 2048$-bit using the fastest known general-purpose algorithm (Number Field Sieve) takes time about $T_{GF}(2048) \approx 2^{103} \cdot T_{DES}$, where T_{DES} denotes the time to perform one encryption operation of the well-known DES symmetric cipher [16]. Assuming conservatively that $T_{CL}(n, \epsilon) \approx 2^{n/4 \cdot \epsilon} \cdot T_{DES}$, we find from the above conjecture that a $(1-\epsilon)n/4$-LSBS RSA modulus achieves the same security level as a random n-bit RSA modulus for $n = 2048$ as long as $n/4 \cdot \epsilon \geq 103$, that is $\epsilon \geq 0.2$, approximately.

4 Partial Key Exposure Attacks on α-LSBS RSA ($\beta > 0$)

In this section we investigate the security of α-LSBS RSA under Partial Key Exposure (PKE) attacks in which $\beta > 0$ LS bits of the secret exponent are known to the attacker. That is, we analyse the security of the (α, β, γ)-LSBS RSA system introduced in the introduction. For all these attacks, the attack input consists of the security parameters (α, β, γ), the α-LSBS modulus N (of length n bit), the public exponent e (with $\log_2 e = \gamma$), and an integer d_0 consisting of the β LS bits of the secret exponent d (that is, $d_0 = d \bmod 2^\beta$). Note we assume that α is known for simplicity but clearly this is no loss of generality since even if α is not made public there are only at most $O(n)$ possible values for it, so in polynomial time we can exhaustively search through this small number of possibilities.

4.1 PKE with Low Public Exponent (Small γ)

4.1.1 Analysis of Generalized Boneh-Durfee-Frankel Attack ($\beta \geq n/4 + \alpha$)

Boneh, Durfee and Frankel (BDF) presented in [5] a PKE attack on low public-exponent RSA when $n/4$ LS bits of the secret exponent d are given. Prior to the results presented in this paper, it was believed that this attack applies in general to all RSA moduli. However, as will be seen below, our work shows that the original result of BDF actually applies only to the specific case of 1-LSBS moduli, and can be stated as follows (following the publication [20] of a preliminary version of our result, the authors of [5] produced a revised version of their paper [6] which contains the statement below).

Theorem 4.1 (BDF). *Given (N, e, d_0), where N is an n-bit 1-LSBS RSA modulus, e is a public exponent (with $\lceil \log_2 e \rceil = \gamma \leq n/4 - 3$) and d_0 consists of the $n/4$ LS bits of the secret exponent d, the BDF attack factors N within time bound $O(\gamma 2^\gamma \cdot T_{Cop}(n))$. Here $T_{Cop}(n)$ denotes the running time of Coppersmith's algorithm of Theorem 2.1.*

In this section we analyze the Boneh-Durfee-Frankel (BDF) PKE attack on low public-exponent RSA [5] in the case of α-LSBS moduli with arbitrary α. Note that the attack is presented and analyzed here in a generalized form, assuming an arbitrary number $\beta \geq n/4$ of exposed secret exponent LS bits, rather than the fixed value $\beta = n/4$ assumed in [5], in order to obtain a success lower bound on β in terms of α. We show that this attack becomes less effective as the sharing parameter α increases. More precisely, following theorem shows that the attack is tractable only when $\beta \geq n/4 + \alpha$. For the case $\beta = n/4$, the running time of the attack for α-LSBS moduli is larger by a factor of about 2^α than the time bound of Theorem 4.1 obtained by BDF for the case of 1-LSBS moduli, which shows that the attack is intractable in this case when α is large.

Theorem 4.2. *Given (N, e, d_0), where N is an n-bit α-LSBS RSA modulus, e is a public exponent (with $\lceil \log_2 e \rceil = \gamma < \beta$) and d_0 consists of the β LS bits of the secret exponent d, the Generalized BDF attack factors N within the following time bound:*

$$T_{BDF}(n) = \begin{cases} O\left(\gamma 2^\gamma \cdot \lceil 2^{n/4 - \beta/2} \rceil \cdot T'_{Cop}(n)\right) & \text{if } \beta < 2(\alpha - 1) + \gamma \\ O\left(\gamma 2^\gamma \cdot \lceil 2^{n/4 + \alpha - \beta} \rceil \cdot T'_{Cop}(n)\right) & \text{if } \beta \geq 2(\alpha - 1) + \gamma \end{cases} \tag{7}$$

Here $T'_{Cop}(n) = T_{Cop}(n) + n^2$ and $T_{Cop}(n)$ denotes the running time of Coppersmith's algorithm of Theorem 2.1.

Proof. The Generalized BDF attack attempts to recover the $n/4$ LS bits of p or q and then use Coppersmith's algorithm of Theorem 2.1 to efficiently factor N. To do this, the attack computes in turn each element of a set $X = \{x_1, ..., x_{|X|}\}$ of trial values for the $n/4$ LS bits of p or q, running Coppersmith's algorithm of Theorem 2.1 to try to factor N with each trial value x_i. The set X is guaranteed by construction to contain p_0 and q_0, the $n/4$ LS bits of p and q respectively. Hence the attack factors N within time bound $O\left(|X| \cdot T'_{Cop}(n)\right)$, where $|X|$ denotes the size of set X and $T'_{Cop}(n)$ is the polynomial running time bound for Coppersmith's algorithm plus the $O(n^2)$ time for the other computations that the attack performs for each execution of Coppersmith's algorithm.

The central part of the attack is the construction of the set X since it must be small enough (i.e. polynomial in n) to make the attack tractable. It is constructed as the set of solutions to a quadratic modular equation as follows. Recall that from the key generation equation (4), we have

$$ed - 1 - k \cdot \phi = 0$$

and

$$\phi = N + 1 - p - N/p.$$

Defining the function f by $f(x) \overset{\text{def}}{=} N + 1 - x - N/x$, we have that $f(p) = f(q) = \phi$ and hence p and q are roots of the quadratic equation

$$(ed - 1) \cdot x - k \cdot xf(x) = 0. \tag{8}$$

Reducing (8) modulo 2^β and using the fact that $d_0 = d \bmod 2^\beta$ is known, we see that $p_0 = p \bmod 2^\beta$ and $q_0 = q \bmod 2^\beta$ are roots of the modular equation:

$$kx^2 + (ed_0 - 1 - k(N+1))x + kN \equiv 0 \pmod{2^\beta}. \tag{9}$$

All the parameters appearing in the coefficients of (9) are known to the attacker with the exception of k. However, since $k = e \cdot (d/\phi) - 1/\phi$, we have from $d < \phi$ that $k < e$ and hence $k \in \{1, \ldots, e-1\}$. When e is small, it is feasible to exhaustively search through all possible candidates $k' \in \{1, \ldots, e-1\}$ for the true value of k. So the attack proceeds as follows. For each candidate $k' \in \{1, ..., e-1\}$ for the true value of k, the attacker forms the candidate modular quadratic equation

$$k'x^2 + (ed_0 - 1 - k'(N+1))x + k'N \equiv 0 \pmod{2^\beta}. \tag{10}$$

Let $\mu(k') \overset{\text{def}}{=} m_2(k')$ denote the multiplicity of 2 in k'. Note that $ed_0 - 1 \equiv ed - 1 \equiv k\phi \pmod{2^\beta}$ implies $ed_0 - 1 \equiv 0 \pmod{2^{m_2(k)+1}}$. Thus the attacker can immediately reject candidates k' for which $ed_0 - 1 \not\equiv 0 \pmod{2^{\mu(k')+1}}$. Otherwise, dividing (10) by $2^{\mu(k')}$ and multiplying by the multiplicative inverse $\theta(k') \overset{\text{def}}{=} \text{odd}(k')^{-1} \bmod 2^{\beta-\mu(k')}$, we have that (10) becomes

$$x^2 + [\theta(k')\frac{ed_0 - 1}{2^{\mu(k')}} - (N+1)]x + N \equiv 0 \pmod{2^{\beta-\mu(k')}}. \tag{11}$$

Let $b(k') \overset{\text{def}}{=} \theta(k')\frac{ed_0-1}{2^{\mu(k')}} - (N+1)$ and $c(k') \overset{\text{def}}{=} N$ denote the known coefficients of (11). By 'completing the square' (using the fact that $b(k')$ is even), we find that (11) is equivalent to

$$[x + b(k')/2]^2 \equiv \Delta(k') \pmod{2^{\beta-\mu(k')}}, \tag{12}$$

where $\Delta(k') \overset{\text{def}}{=} (b(k')/2)^2 - c(k')$. Note that if the candidate k' is correct so $k' = k$ then, using $ed_0 - 1 \equiv ed - 1 \equiv k(N + 1 - p - q) \pmod{2^\beta}$, the linear coefficient of (11) reduces to $b(k) \equiv \theta(k)\frac{k(N+1-(p+q))}{2^{\mu(k)}} - (N+1) \equiv -(p+q) \pmod{2^{\beta-\mu(k)}}$ and hence

$$\Delta(k) \equiv (b(k)/2)^2 - c(k) = \left(\frac{p+q}{2}\right)^2 - N = \left(\frac{p-q}{2}\right)^2 \pmod{2^{\beta-\mu(k)}}.$$

Since $m_2(p-q) = \alpha$, it follows that if $k' = k$ then there are two subcases. In the case $2(\alpha-1) \geq \beta - \mu(k)$, we have $\Delta(k) \equiv 0 \pmod{2^{\beta-\mu(k)}}$. In the case $2(\alpha-1) < \beta-\mu(k)$, we have $m_2(\Delta(k)) = 2(\alpha-1)$. Accordingly, for each candidate $k' \in \{1, \dots, e-1\}$, the attacker solves (12) by applying the efficient modular square-root algorithm of Lemma 2.1.

In the case $2(\alpha-1) \geq \beta-\mu(k')$, the attacker knows that $\Delta(k') \equiv 0 \pmod{2^{\beta-\mu(k')}}$ (else k' cannot be equal to k and is immediately rejected), and obtains from part (2.1) of Lemma 2.1 that $x+b(k')/2 \equiv 0 \pmod{2^{\lceil(\beta-\mu(k'))/2\rceil}}$. In this case the attacker knows $\lceil(\beta-\mu(k'))/2\rceil$ LS bits of x and needs to exhaustively search the missing $\max(0, n/4 - \lceil(\beta-\mu(k'))/2\rceil)$ MS bits to get the $n/4$ LS bits necessary to apply Coppersmith's algorithm. Thus testing k' in this case requires testing $|X_1(k')|$ candidates for the $n/4$ LS bits of p, where

$$|X_1(k')| = \left\lceil 2^{n/4-\lceil(\beta-\mu(k'))/2\rceil} \right\rceil \leq \left\lceil 2^{\mu(k')/2} \cdot 2^{n/4-\beta/2} \right\rceil. \tag{13}$$

In the case $2(\alpha-1) < \beta-\mu(k')$, the attacker has $m_2(\Delta(k')) = 2(\alpha-1)$ (else k' cannot be equal to k and is immediately rejected), and obtains from part (2.2) of Lemma 2.1 in time $O(n^2)$ at most four candidates for λ such that $x + b(k')/2 \equiv \lambda \pmod{2^{\beta-\mu(k')-(\alpha-1)}}$. In this case the attacker has at most 4 candidates for the $\beta-\mu(k')-(\alpha-1)$ LS bits of x and needs to exhaustively search the missing $\max(0, n/4-(\beta-\mu(k')-(\alpha-1)))$ MS bits to get the $n/4$ LS bits necessary to apply Coppersmith's algorithm. Thus testing k' in this case requires testing $|X_2(k')|$ possible values for the $n/4$ LS bits of p, where

$$|X_2(k')| \leq \left\lceil 2^{\mu(k')+1} \cdot 2^{n/4+\alpha-\beta} \right\rceil. \tag{14}$$

Note that if $\beta < 2(\alpha-1)+\gamma$ then both $|X_1(k')|$ and $|X_2(k')|$ are upper bounded by $\left\lceil 4 \cdot 2^{\mu(k')} \cdot 2^{n/4-\beta/2} \right\rceil$ for all $k' \in \{1, \dots, e-1\}$ and if $\beta \geq 2(\alpha-1)+\gamma$ then bound $|X_2(k')|$ applies for all $k' \in \{1, \dots, e-1\}$. Hence, summing over all $k' \in \{1, \dots, e-1\}$, the total number $|X|$ of possible values for the $n/4$ LS bits of p that need to be tested is bounded as:

$$|X| = \begin{cases} O\left(\sum_{k'=1}^{e-1} \left\lceil 2^{\mu(k')} 2^{n/4-\beta/2} \right\rceil\right) & \text{if } \beta < 2(\alpha-1)+\gamma \\ O\left(\sum_{k'=1}^{e-1} \left\lceil 2^{\mu(k')} 2^{n/4+\alpha-\beta} \right\rceil\right) & \text{if } \beta \geq 2(\alpha-1)+\gamma \end{cases} \tag{15}$$

The claimed running-time $T_{BDF}(n) = O\left(|X| \cdot T'_{Cop}(n)\right)$ now follows immediately from the fact that, since $\gamma = \lceil \log_2 e \rceil$,

$$\sum_{k'=1}^{e-1} 2^{\mu(k')} = \sum_{m=0}^{\gamma-1} 2^m |H(m)| \leq \sum_{m=0}^{\gamma-1} 2^m \lfloor e/2^m \rfloor \leq \gamma 2^\gamma,$$

where $H(m)$ denotes the set of all $k' \in \{1, \dots, e-1\}$ such that $\mu(k') = m$. This completes the proof of the theorem. □

□

As explained in Section 3, when the prime factors p and q of N are chosen randomly and indpendently, α is a small constant with high probability and so in that case our result reduces to that obtained by BDF, namely that $\beta \geq n/4$ suffices.

4.1.2 An intractability result for $\beta \leq 2\alpha$

In the previous section we showed that the BDF partial key exposure attack on low public-exponent α-LSBS RSA is tractable only if $\beta \geq n/4 + \alpha$ LS bits of d are available, rather than the bound $\beta \geq n/4$ shown by BDF [5] for small α. However, the BDF attack is only one specific attack, and it is natural to ask whether it is possible to modify it or find another attack which works efficiently for $\beta \geq n/4$ even for large α. The following result shows that the existence of such an attack for $\alpha \geq n/8$ would imply the existence of a polynomial-time factoring algorithm for α-LSBS moduli with $\alpha \geq n/8$ (recall from Section 3 that the best known polynomial-time factoring algorithm for α-LSBS moduli works only when $\alpha \geq n/4$).

Theorem 4.3. *Suppose there exists a PKE attack algorithm* A *that, given* (N, e, d_0)*, factors* N *in time* $T_A(n)$*, where* N *is an* n*-bit* α*-LSBS RSA modulus,* e *is a public exponent (with* $\lceil \log_2 e \rceil = \gamma$*) and* d_0 *consists of the* $\beta \leq 2\alpha$ *LS bits of the secret exponent* d*. Then there exists a factoring algorithm* F *for* α*-LSBS moduli, that given only* (N, e)*, factors* N *in time* $T_F(n)$*, with*

$$T_F(n) = O\left(2^\gamma \cdot (T_A(n) + n^2)\right).$$

Proof. We show how to construct the factoring algorithm F. Given (N, e), F simply computes in time $O(2^\gamma \cdot n^2)$ a set of $4 \cdot 2^\gamma$ candidates d'_0 for d_0 which is guaranteed to contain the correct d_0, the β LS bits of d, and runs A on input (N, e, d'_0) for each candidate d'_0 for d_0. Thus F succeeds to factor N in time $T_F(n) \leq O\left(2^\gamma \cdot (T_A(n) + n^2)\right)$, as claimed. To find the $4 \cdot 2^\gamma$ candidates for d_0, F does the following. First, F applies the algorithm of Lemma 3.1 to compute in time $O(n^2)$ up to 4 candidates for s_0 such that $s_0 \equiv p + q \pmod{2^{2\alpha}}$. Now note that reducing the key generation equation $ed - 1 = k(N + 1 - (p + q))$ modulo $2^{2\alpha}$ gives $ed - 1 \equiv k(N + 1 - s_0) \pmod{2^{2\alpha}}$. Since e is odd, it has a multiplicative inverse e^{-1} modulo $2^{2\alpha}$ which can be computed in time $O(n^2)$ and we have that

$$d \equiv e^{-1} \cdot [1 + k(N + 1 - s_0)] \pmod{2^{2\alpha}}.$$

As in the proof of Theorem 4.2, F knows that $k \in \{1, \ldots, e-1\}$. So, for each candidate k' $\{1, \ldots, e-1\}$ for k and each of the four candidates s'_0 for s_0, F computes a corresponding candidate $d'_0 = e^{-1} \cdot [1 + k'(N + 1 - s'_0)] \bmod 2^{2\alpha}$ for $d \bmod 2^{2\alpha}$, and hence (using $\beta \leq 2\alpha$), F obtains that $d'_0 \bmod 2^\beta = d_0$ for one of the $4(e - 1) \leq 4 \cdot 2^\gamma$ candidates for d'_0, as required. This completes the proof. □

□

The proof of the above result shows more generally, that for low public exponents, α-LSbS RSA 'leaks' the 2α LS bits of the secret exponent d. Therefore, for $\beta \leq 2\alpha$ and small γ, the one-wayness of (α, β, γ)-LSBS RSA (PKE with β LS bits of d given) is equivalent to the one-wayness of $(\alpha, 0, \gamma)$-LSBS RSA (no bits of d given), which in turn, as explained in Section 3, is conjectured to be equivalent to the problem of factoring α-LSBS moduli. We conclude that when $\beta \leq 2\alpha$ and $\alpha = (1 - \epsilon)n/4$ for some constant $\epsilon > 0$, then (α, β, γ)-LSBS RSA appears asymptotically as secure as standard RSA, but has potential efficiency advantages, as illustrated in the proposed application in Section 5.

The result in this section will be used in Section 5 in conjunction with the following result of Boneh, Durfee and Frankel [5].

Theorem 4.4 (BDF[5]). *Suppose there exists a PKE attack algorithm* A *that, given* (N, e, d_0)*, factors* N *in time* $T_A(n)$*, where* N *is an* n*-bit RSA modulus,* e *is a public exponent (with* $\lceil \log_2 e \rceil = \gamma$*) and* d_0 *consists of the* $n/2$ *MS-bits of the secret exponent* d*. Then there exists a factoring algorithm* F*, that given only* (N, e)*, factors* N *in time* $T_F(n)$*, with*

$$T_F(n) = O\left(2^\gamma T_A(n)\right).$$

This result shows the for small public exponents, the $n/2$ MS bits of d are also 'leaked' from the public information (N, e).

4.2 PKE with Large Public Exponent ($\gamma \geq n/6$)

All the results of Section 4.1 are useful only for the case of a *small* public exponent e because of the factor 2^γ in the attack/reduction running times. In particular, the reduction of Theorem 4.3 also deteriorates by a factor of 2^γ and hence it does not rule out the possibility that when α and γ are large, there may be an efficient PKE attack using $\beta < 2\alpha$ LS bits of d. Indeed, in this section we show improved attacks which confirm this possibility.

4.2.1 First Attack ($\alpha \geq n/8$)

In the first attack, we assume that $\alpha \geq n/8$. We modify the Generalized BDF attack of Theorem 4.2 in two ways.

First, we use the assumption that α is large to eliminate the exhaustive search over 2^γ elements required to find k by computing k modulo $2^{\min(2\alpha,\beta)-m_2(\phi)}$. This involves combining knowledge of $s_0 = s \bmod 2^{2\alpha}$, the 2α LS bits of $s = p + q$ (from Lemma 3.1 part 2) and the β LS bits of d. Thus if $\min(2\alpha, \beta) - m_2(\phi) > \gamma$, this gives us all the bits of k.

Second, once we know k, we use it to compute $s \bmod e$ by reducing the key generation equation modulo e (note that this trick was also used in an attack with known *Most Significant* bits of d in [5] to eliminate d once k is known), which allows us to compute $s_1 = s \bmod e$. Now we know both $s \bmod e$ and $s \bmod 2^{2\alpha}$ and we use the Chinese Remainder Theorem (CRT) to combine them and compute $s_2 = s \bmod e2^{2\alpha}$. Thus if $\gamma + 2\alpha \geq n/2$ we have all of s and we can then easily factor N as in Corollary 3.1.

The precise statement and proof of this result follows.

Theorem 4.5. *Given (N, e, d_0), where N is an n-bit α-LSBS RSA modulus, e is a public exponent (with $\lceil \log_2 e \rceil = \gamma \leq \min(2\alpha, \beta) - m_2(\phi)$, where $m_2(\phi) < \min(2\alpha, \beta)$ denotes the multiplicity of 2 in $\phi = N + 1 - (p + q)$) and d_0 consists of the β LS bits of the secret exponent d, there exists an attack which factors N within time bound $O(\lceil 2^{n/2-2\alpha-\gamma} \rceil\, p(n))$, where $p(n)$ is a polynomial in n.*

Proof. The attacker knows $d_0 = d \bmod 2^\beta$ and, by applying Lemma 3.1 computes at most 4 candidates for $s_0 = s \bmod 2^{2\alpha}$ in time $O(n^2)$, where $s = p + q$. Hence, from the key generation equation $ed - 1 = k\phi = k(N + 1 - s)$, the attacker obtains the congruence

$$ed_0 - 1 \equiv k(N + 1 - s_0) \pmod{2^{\min(2\alpha,\beta)}}. \tag{16}$$

Since $N+1-s_0 \equiv \phi \pmod{2^{\min(2\alpha,\beta)}}$ and $m_2(\phi) < \min(2\alpha, \beta)$, it follows that $m_2(N+1-s_0) = m_2(\phi)$. Thus $N+1-s_0 = \lambda \cdot 2^{m_2(\phi)}$ for some odd λ. By multiplying the congruence (16) by the multiplicative inverse λ^{-1} of λ modulo $2^{\min(2\alpha,\beta)}$ and dividing by $2^{m_2(\phi)}$ the attacker obtains that

$$k \equiv \lambda^{-1} \cdot \frac{ed_0 - 1}{2^{m_2(\phi)}} \pmod{2^{\min(2\alpha,\beta)-m_2(\phi)}}. \tag{17}$$

Using (17) the attacker therefore obtains in time $O(n^2)$ four candidates for $k_0 \overset{\text{def}}{=} k \bmod 2^{\min(2\alpha,\beta)-m_2(\phi)}$, and since $k < e \leq 2^\gamma \leq 2^{\min(2\alpha,\beta)-m_2(\phi)}$, it follows that one of the four candidates for k_0 is equal to k.

Now the attacker has 4 candidates for k. Reducing the key generation equation modulo e gives that $k \cdot (s - (N + 1)) \equiv 1 \pmod{e}$. Therefore (noting that $\gcd(k, e) = 1$),

$$s \equiv k^{-1} + N + 1 \pmod{e}$$

and using this the attacker computes 4 candidates for $s_1 \overset{\text{def}}{=} s \bmod e$ in time $O(n^2)$. So the attacker obtains 4 candidate pairs for (s_0, s_1) where $s_0 = s \bmod 2^{2\alpha}$ and $s_1 = s \bmod e$. Since $\gcd(2^{2\alpha}, e) = 1$, the attacker can use the Chinese Remainder Theorem to combine these two 'portions' of s and computes $s_2 \overset{\text{def}}{=} s \bmod 2^{2\alpha}e$. Since s_2 is the remainder of s on division by $2^{2\alpha}e$ we have $s = u_2 \cdot 2^{2\alpha}e + s_2$, where the positive quotient integer u_2 is upper bounded as

$$u_2 \leq s/(2^{2\alpha}e) \leq 8 \cdot 2^{n/2-(2\alpha+\gamma)}.$$

So, for each of the four candidates for s_2 the attacker tries all possible $\lceil 8 \cdot 2^{n/2-(2\alpha+\gamma)} \rceil$ candidates for u_2 to get $O(\lceil 2^{n/2-(2\alpha+\gamma)} \rceil)$ candidates for $s = u_2 \cdot 2^{2\alpha}e + s_2$. For each of these candidates for s, the attacker tries to find p by solving the quadratic

$$x^2 - s \cdot x + N = 0,$$

and testing whether the solution divides N. When the correct value of s is reached, the factor p of N will be found, after at most a running time of $O(\lceil 2^{n/2-(2\alpha+\gamma)} \rceil\, p(n))$, where $p(n)$ is the polynomial running time of all computations involved in computing and testing each candidate for p. This establishes the claimed running time for the attack and completes the proof. □

□

Summarizing, this attack is tractable when $n/2 - 2\alpha \leq \gamma \leq \min(\beta, 2\alpha) - m_2(\phi)$, which shows that we must have $\alpha \geq n/8$ (note that $m_2(\phi)$ is a small constant with high probability for a random α-LSBS modulus). For example, for $\alpha = n/8$ and $\gamma = n/4$, it suffices to have $\beta \geq n/4$ LS bits of d, the same bound for β as obtained by BDF for $\alpha = 1$ and γ small. For even larger $\alpha = (1-\epsilon)n/4$ for a constant $\epsilon < 1/2$, we only need γ and β to exceed $\epsilon \cdot n/2$ for the attack to be tractable, much less than the $2\alpha = (1-\epsilon)n/2$ bits lower bound on β of Theorem 4.3, which holds for small (constant) γ.

4.2.2 Second Attack (e prime, $\alpha \geq n/12$)

The second attack is a variant of the first which is tractable for even lower α, namely $\alpha \geq n/12$. However, it also assumes that e is a prime. The first part of the attack involves computing four candidates for k (under the assumption that $\gamma \leq \min(2\alpha, \beta) - m_2(\phi)$) and hence four candidates for $s_1 = s \bmod e$ exactly as in the proof of Theorem 4.5. But now, instead of using s_1 to compute $s_2 = s \bmod 2^{2\alpha}e$, we use it to compute $p_1 = p \bmod e$ by solving a quadratic modulo e (note that this process of obtaining $p \bmod e$ from $s \bmod e$ has also has been in used in an attack in [5] given MS bits of d). Then we use Chinese Remaindering together with Lemma 3.1 to get $p_2 = p \bmod 2^{\alpha}e$ and try to factor N using p_2 and Coppersmith's algorithm of Theorem 2.1.

The precise statement and proof of this result follows.

Theorem 4.6. *Given (N, e, d_0), where N is an n-bit α-LSBS RSA modulus, e is a prime public exponent (with $\lceil \log_2 e \rceil = \gamma \leq \min(2\alpha, \beta) - m_2(\phi)$, where $m_2(\phi) < \min(2\alpha, \beta)$ denotes the multiplicity of 2 in $\phi = N + 1 - (p+q)$) and d_0 consists of the β LS bits of the secret exponent d, there exists an attack which factors N within time bound $O(\lceil 2^{n/4-\alpha-\gamma} \rceil\, p(n))$, where $p(n)$ is a polynomial in n.*

Proof. The attacker computes four candidates for $s_1 = s \bmod e$ in time polynomial in n exactly as in the proof of Theorem 4.5. We know that $p_1 = p \bmod e$ is a solution to the quadratic congruence

$$x^2 - s_1 \cdot x + N = 0 \pmod{e}. \tag{18}$$

Since e is prime the the attacker can solve the congruence (18) in randomized polynomial time (see [13]) to obtain 2 candidates for p_1 for each of the 4 candidates for s_1. Also, by applying Part 1 of Lemma 3.1, the attacker computes 4 candidates for $p_0 = p \bmod 2^{\alpha}$. Since $\gcd(2^{\alpha}, e) = 1$, the attacker can use the Chinese Remainder Theorem to combine these two 'portions' of p and computes 32 candidates for $p_2 \overset{\text{def}}{=} p \bmod 2^{\alpha}e$. If $\alpha + \gamma - 1 \geq n/4$ then it possible to factor N using p_2 and Coppersmith's algorithm of Theorem 2.1. Otherwise, let $p_3 = p \bmod 2^{n/4-\gamma+1}e$. Note that $2^{n/4-\gamma+1}e \geq 2^{n/4}$ and hence it is possible to factor N given p_3 using Theorem 2.1. But p_2 is the remainder of p_3 on division by $2^{\alpha}e$ and we have $p_3 = u_2 \cdot 2^{\alpha}e + p_2$, where the positive quotient integer u_2 is upper bounded as

$$u_2 \leq p_3/(2^{\alpha}e) \leq 4 \cdot 2^{n/4-(\alpha+\gamma)}.$$

So, for each of the 32 candidates for p_2 the attacker tries all possible $4 \cdot 2^{n/4-(\alpha+\gamma)}$ candidates for u_2 to get $128 \cdot 2^{n/4-(\alpha+\gamma)}$ candidates for p_3. For each of these candidates for p_3, the attacker runs the algorithm of Theorem 2.1 to try to factor N. Thus N is factored after at most a running time of $128 \cdot 2^{n/4-(\alpha+\gamma)}p(n)$, where $p(n)$ is the polynomial running time of all computations involved in computing and testing each candidate for p_3. This establishes the claimed run time for the attack and completes the proof. □

□

In summary, this attack is tractable when $n/4 - \alpha \leq \gamma \leq \min(\beta, 2\alpha) - m_2(\phi)$, which shows that we must have $\alpha \geq n/12$. For example, for $\alpha = n/12$ and $\gamma = n/6$, it suffices to have $\beta \geq n/6$ LS bits of d, which is smaller than the bound $\beta \geq n/4$ obtained by BDF for $\alpha = 1$ and γ small. For even larger $\alpha = (1-\epsilon)n/4$ for a constant $\epsilon < 1/2$, we only need γ and β to exceed $\epsilon \cdot n/4$. Finally, we note that, as explored in [5], the requirement for this attack that e is prime can be relaxed under some conditions, e.g. when e has a small number of prime factors which are all known. This is because the only requirement of the attack is that all solutions of the quadratic (18) can be found in polynomial time.

5 Application to Server-Aided Signature Generation

The result of Theorem 4.3 in Section 4.1.2 that α-LSBS RSA with low public exponents 'leaks' the 2α LS bits of the secret exponent d, together with the result of Boneh, Durfee and Frankel (see Theorem 4.4 in Section 4.1.2) that the $n/2$ MS bits of d are also leaked for low public-exponents, opens the possibility of fast RSA Server-Aided Decryption or Signature Generation. In particular, if $\alpha = (1-\epsilon)n/4$, this means that one can reveal to the public server the majority of bits of d except for the block of about $n/2 - 2\alpha = \epsilon \cdot n/2$ 'middle' bits in positions $n/2 - 1$ down to 2α. By offloading the exponentiation portion corresponding to the public bits of d to the fast server, the card's computation can be significantly reduced, while still provably achieving the same security as the one-wayness of α-LSBS RSA.

We give the details of the scheme below.

5.1 Definition of SASG

We use the following general definition for SASG. Given a digital signature scheme $\mathsf{DS} = (\mathsf{GK}, \mathsf{S}, \mathsf{V})$ with key-pair generation algorithm GK, signature generation algorithm S and signature verification algorithm V, a Server-Aided Signature Generation (SASG) protocol consists of two interacting algorithms: (1) A signature generation card with input the signing secret key sk, and (2) A fast aiding server with input public key pk. The common input to both card and server is the message to be signed M. At the end of the protocol between card and server, the card outputs a signature σ for the scheme DS on message M with respect to the secret key sk. The pair (M, σ) can then be publicly verified with respect to the public key pk using the normal verification algorithm V for scheme DS.

Efficiency. The *computational efficiency* of a SASG protocol is defined, naturally, the ratio by which the computation time of card is smaller than the computation time of S, the standard signature generation algorithm for the scheme DS. The *communication overhead* of a SASG protocol is total bit-length of protocol messages exchanged between card and server in the protocol (this does not include the length of M and σ).

Security. We can define two types of security of a SASG protocol based on the standard model of existential unforgeability under adaptive chosen message attacks for digital signature schemes [9]. The protocol is said to be CMA-Secure against *Passive-Server* attack if it is infeasible for an attacker to produce an existential forgery on a 'new message' after polynomially many (in the security parameter k of the scheme) protocol runs in which the attacker interacts with the card as a server with attacker-chosen messages, but assuming that the attacker follows the protocol in sending all protocol messages to the card (if the protocol is secure even when attacker is also allowed to deviate from the protocol when sending messages to the card, we say that it is CMA-Secure against *Active-Server* attacks).

5.2 The Protocol

As mentioned the idea of the protocol is simple: we reveal the leaked bits 2α LS bits and $n/2$ MS bits of d to the server. The underlying digital signature scheme can be any RSA-based one, but here we use the classical $\mathsf{FDH}-\mathsf{RSA}$ 'Full Domain Hash' scheme [2].

Let $H(.) : \{0,1\}^* \to \mathbb{Z}_N^*$ be cryptographic hash-function. The key generation algorithm GK generates a public key (N, e, α, d_{pub}) and a secret key (N, e, α, d_{sec}) with N an α-LSBS RSA modulus, e a small public exponent, and d_{pub} and d_{sec} consist of the public and secret portions of the secret exponent $d = e^{-1} \bmod \phi$, respectively. That is, write the binary representation of the secret exponent as $d = \sum_{i=0}^{n-1} d_i 2^i$, where $d_i \in \{0,1\}$ represents the i'th significant bit of d. Then $d_{sec} \overset{\text{def}}{=} (1/2^{2\alpha}) \cdot \sum_{i=2\alpha}^{n/2-1} d_i 2^i$ and $d_{pub} \overset{\text{def}}{=} d - 2^{2\alpha} d_{sec}$. On common input M, the SASG protocol for computing the signature $\sigma = H(M)^d \bmod N$ consists of the following two steps:

(1) The server computes $\beta_1 \overset{\text{def}}{=} H(M)^{d_{pub}} \bmod N$ and $\beta_2 \overset{\text{def}}{=} H(M)^{2^{2\alpha}} \bmod$ and forwards the pair (β_1, β_2) to the card.

(2) The card outputs the signature $\sigma \overset{\text{def}}{=} \beta_1 \beta_2^{d_{sec}} \bmod N$.

5.2.1 Computational Efficiency

Suppose $\alpha = (1-\epsilon)n/4$. The length of the exponent d_{sec} in the exponentiation performed by the card is only $n/2 - 2\alpha = \epsilon \cdot n/2$ bits, compared to the length of d, namely n bits, used in the standard signing algorithm S. Since exponentiation time is linear in the length of the exponent, the computational cost for the card is reduced by a factor of around $2/\epsilon$, which can be very significant, especially for $\epsilon < 1/2$. For example, if we use the conjecture in Section 3 on the difficulty of factoring $(1-\epsilon)n/4$-LSBS RSA moduli, we have that for $n = 2048$-bit moduli, ϵ can be made as small as 0.2 (approximately) while leaving the security unchanged from that of random RSA moduli of length $n = 2048$ bit. Thus at this level of security we can achieve a computational efficiency ratio of up to $2/0.2 = 10$. We remark that our technique is also useful when the Chinese Remainder Theorem is used to speed-up computation by the card (see, for example [15]), achieving in these cases a computation saving for the card by a factor of around $1/\epsilon$. The communication overhead of the protocol is $2n$ bits.

5.2.2 Security

In the Passive-Server attack setting, the only difference between the view of the attacker in the SASG protocol to its view in the CMA attack on $\mathsf{FDH}-\mathsf{RSA}$ is that in the former case the public key contains d_{pub}. But since $\beta \leq 2\alpha$ and e is small, it follows from Theorems 4.3 and 4.4 that these bits are 'leaked' by (N, e). Hence it follows that the protocol is CMA-secure against Passive-Server attacks as long as the underlying $\mathsf{FDH}-\mathsf{RSA}$ signature is CMA-secure, which follows in turn in the random oracle model [2] from the one-wayness of the RSA primitive, which in our case is the α-LSBS RSA assumption. Note that the protocol is not designed for security against active servers.

5.3 Practical Generation of α-LSBS RSA Moduli

In practice, generating α-LSBS RSA moduli in the natural way, i.e. picking one of the primes (say p) randomly, and then testing candidate integers for q of the form $q = (p+2^{\alpha}) \bmod 2^{\alpha+1} + r \cdot 2^{\alpha+1}$ (with a randomly chosen r) for primality, is asymptotically expected to be as efficient as the 'standard' independent primes generation algorithm for random RSA moduli, where each candidate for q is chosen independently of p as a random odd integer. This is due to a quantitative version of Dirichlet's Theorem (see [18]), which implies that the density of primes less than a bound x in *any* arithmetic progression $q \equiv a \pmod{z}$ (with $\gcd(a, z) = 1$) converges to $(z/\phi(z)) \cdot (1/\ln x)$. For the case $z = 2^{\alpha+1}$, we have $2^{\alpha+1}/\phi(2^{\alpha+1}) = 2$ for all $\alpha \geq 1$. Therefore, the density of primes converges to $2/\ln x$ for both the standard modulus generation search (where $a = 1$ and $z = 2$), as well as the α-LSBS modulus generation search (where $a = (p + 2^{\alpha}) \bmod 2^{\alpha+1}$ and $z = 2^{\alpha+1}$).

6 Conclusion

We investigated the security of RSA under partial key exposure attacks for α-LSBS moduli, in which the modulus prime factors share α least significant bits. We have also suggested an application for α-LSBS RSA in fast public-server-aided RSA signature generation. It was shown that the saving in computation using this technique is in the order of $1/\epsilon$ if α-LSBS moduli with $\alpha = (1-\epsilon)n/4$ are used. This motivates further study of the complexity of factoring such RSA moduli for small ϵ. The aim of such study would be either to strengthen confidence in the current conjecture that a time of $\min(T_{GF}(n), O(2^{\epsilon \cdot n/4}))$ is required to factor $(1-\epsilon)n/4$-LSBS moduli (where $T_{GF}(n)$ is the run-time of the best general-purpose factoring algorithm for n-bit RSA moduli), or to find a new factoring algorithm for $(1-\epsilon)n/4$-LSBS moduli which works in polynomial time even for some constant $\epsilon > 0$. In any case this would improve confidence in the limit on the effectiveness of the technique.

Acknowledgements. The authors would like to thank the anonymous referees of CT-RSA 2001 for their helpful comments on a preliminary version [20] of some of the results in this paper.

References

[1] E. Bach and J. Shallit. *Algorithmic Number Theory, Vol. I.* MIT Press, Massachusetts, 1996.

[2] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In *EUROCRYPT '96*, volume 1070 of *LNCS*, pages 399–416, Berlin, 1996. Springer-Verlag.

[3] J. Blömer and A. May. New Partial Key Exposure Attacks on RSA. In *Crypto 2003*, LNCS, pages 27–43. Springer-Verlag, 2003.

[4] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the American Mathematical Society (AMS)*, 46(2):203–213, 1999.

[5] D. Boneh, G. Durfee, and Y. Frankel. An Attack on RSA Given a Small Fraction of the Private Key Bits. In *ASIACRYPT '98*, volume 1514 of *LNCS*, pages 25–34, Berlin, 1998. Springer-Verlag.

[6] D. Boneh, G. Durfee, and Y. Frankel. Exposing an RSA Private Key Given a Small Fraction of its Bits. Available from author's webpage at `http://www2.parc.com/csl/members/gdurfee/pubs.htm`, 2002. Revised version of Asiacrypt '98 paper.

[7] D. Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. of Cryptology*, 10:233–260, 1997.

[8] B. de Weger. Cryptanalysis of RSA with Small Prime Difference. *Applicable Algebra in Engineering, Communication and Computing*, 13:17–28, 2002.

[9] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure against Adaptively Chosen Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[10] M.J. Hinek, M.K. Low, and E. Teske. On some Attacks on Multi-Prime RSA. Cryptology ePrint Archive, Report 2002/063, 2002. `http://eprint.iacr.org/`.

[11] A. Lenstra. Generating RSA Moduli with a Predetermined Portion. In *ASIACRYPT '98*, volume 1514 of *LNCS*, pages 1–10, Berlin, 1998. Springer-Verlag.

[12] A.K. Lenstra. Unbelievable Security: Matching AES Security Using Public Key Systems. In *Asiacrypt 2001*, volume 2248 of *LNCS*, pages 67–86, Berlin, 2001. Springer-Verlag.

[13] R. Lidl and H. Niederreiter. *Finite Fields.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1997.

[14] T. Matsumoto, K. Kato, and H. Imai. Speeding Up Secret Computations with Insecure Auxiliary Devices. In *CRYPTO '88*, volume 403 of *LNCS*, pages 497–506, Berlin, 1989. Springer-Verlag.

[15] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography.* Discrete mathematics and its applications. CRC Press, 1997.

[16] National Bureau of Standards. *Data Encryption Standard, Federal Information Processing Standards Publication 46-2*, 1993.

[17] P. Nguyen and J. Stern. The Béguin-Quisquater Server-Aided RSA Protocol from Crypto '95 is not secure. In *ASIACRYPT '98*, volume 1514 of *LNCS*, pages 372–379, Berlin, 1998. Springer-Verlag.

[18] D. Redmond. *Number Theory: an introduction.* Number 201 in Monographs and textbooks in pure and applied mathematics. Marcel Dekker, 1996.

[19] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–128, 1978.

[20] R. Steinfeld and Y. Zheng. An Advantage of Low-Exponent RSA with Modulus Primes Sharing Least Significant Bits. In *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *LNCS*, pages 52–62, Berlin, 2001. Springer-Verlag.