An Advantage of Low-Exponent RSA with Modulus Primes Sharing Least Significant Bits

Ron Steinfeld and Yuliang Zheng

Laboratory for Information and Network Security,
School of Network Computing,
Monash University,
Frankston 3199, Australia
{ron.steinfeld,yuliang.zheng}@infotech.monash.edu.au

Abstract. Let $N = pq$ denote an RSA modulus of length n bits. Call N an $(m - LSbS)$ RSA modulus if p and q have exactly m equal Least Significant (LS) bits . In Asiacrypt '98, Boneh, Durfee and Frankel (BDF) described several interesting 'partial key exposure' attacks on the RSA system. In particular, for low public exponent RSA, they show how to recover in time polynomial in n the whole secret-exponent d given only the $n/4$ LS bits of d. In this note, we relax a hidden assumption in the running time estimate presented by BDF for this attack. We show that the running time estimated by BDF for their attack is too low for $(m - LSbS)$ RSA moduli by a factor in the order of 2^m. Thus the BDF attack is intractable for such moduli with large m. Furthermore, we prove a general related result, namely that if low-exponent RSA using an $(m - LSbS)$ modulus is secure against poly-time conventional attacks, then it is also secure against poly-time partial key exposure attacks accessing up to $2m$ LS bits of d. Therefore, if low-exponent RSA using $(n/4(1-\epsilon) - LSbS)$ moduli for small ϵ is secure, then this result (together with BDF's result on securely leaking the $n/2$ MS bits of d) opens the possibility of fast and secure public-server-aided RSA decryption/signature generation.

1 Introduction

Let $N = pq$ denote an RSA modulus of length n bits, with p and q primes each of length about $n/2$ bits. In this paper we restrict our attention to *low public exponent* variants of the RSA public key system [11]. For these variants the public exponent e is chosen to be a small value (e.g. 3), independent of the modulus length n. Then a user generates an RSA modulus N and computes his secret exponent d to satisfy $ed = 1 \bmod \phi(N)$, where $\phi(N) = N + 1 - (p + q)$ is Euler's phi function evaluated at N. When used properly, low public exponent RSA (which we hereafter refer to simply as low exponent RSA) is currently considered secure and in fact is in wide use because the encryption operation $x \mapsto x^e \bmod N$ can be performed very quickly, i.e. in time quadratic rather than cubic in n. However, the decryption operation $x \mapsto x^d \bmod N$ still needs cubic time in n and 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 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 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 [5]. Many such schemes have been proposed, but many have been shown to be insecure (see [7] for a recent example).

In AsiaCrypt '98, Boneh, Durfee and Frankel (BDF) described several interesting partial key exposure attacks on the RSA system [1]. In particular, for low exponent RSA, they show 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. They also showed the useful result that knowing the $n/2$ *most* significant (MS) bits of d cannot help an attacker if low-exponent RSA is secure (because these bits are 'leaked' out by the public information). In the context of SASC, these bits can therefore be made available to the public server, which can perform half the decryption exponentiation computation. This gives a reduction by a factor of 2 of the computation performed by the card (possessing the LS bits of d), compared with the unaided case when the card performs the standard exponentiation with the full-length d (from hereon all computation saving factors will be stated with respect to this full-length exponentiation case). However, in cases where the card is able to store the prime factors of N, the Chinese Remainder Theorem (CRT) can be used to reduce the decryption computation by a factor of 4 without any server aid (see, for example [6], section 14.75). When CRT is used by the card, the BDF server-aided technique does not achieve additional savings (i.e. also gives a reduction by a factor of 4) and hence is not useful in these cases.

In this note, we relax a hidden assumption in the running time estimate presented by BDF for their low public exponent key exposure attack (our comments do not apply to the other attacks presented by BDF for large public exponents). Call $N = pq$ an *m-LS bit Symmetric* (or $(m - LSbS)$ for short) RSA modulus, if p and q are primes having exactly m equal LS bits, i.e. $p - q = r \cdot 2^m$ for some odd integer r. We show that the running time estimated by BDF for their attack is too low for $(m - LSbS)$ RSA moduli by a factor in the order of 2^m. Thus the BDF attack is intractable for such moduli if m increases proportionally with n. Furthermore, we prove a general result on $(m - LSbS)$ RSA moduli which can have applications in fast RSA SASC, namely that if a low-exponent RSA system using an $(m - LSbS)$ RSA modulus is secure against arbitrary poly-time 'conventional' attackers (i.e. attackers having *no* access to secret bits), then the system is also secure against arbitrary poly-time partial key exposure attackers having access to up to $2m$ LS bits of the secret exponent d.

Therefore, if low-exponent RSA systems using $(n/4(1-\epsilon) - LSbS)$ moduli with small ϵ are secure (implying in particular that $(n/4(1-\epsilon) - LSbS)$ moduli are hard to factor), then our result, together with BDF's result on securely leaking the $n/2$ MS bits of d, opens the possibility of fast and secure RSA

SASC for decryption or signature generation. In particular, this means that one can reveal to the public server the majority of bits of d except for the block of about $n/2 - 2m = (n/2)\epsilon$ 'middle' bits in positions $n/2 - 1$ down to $2m$. 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 $\epsilon < 1/2$. Unlike the BDF case, this technique is also useful when CRT is used by the card, achieving in these cases a computation saving for the card by a factor $4/\epsilon$.

2 Review of Boneh-Durfee-Frankel Attack

In this section we review the BDF partial key exposure attack on low public exponent RSA. The attack can be simply described as it relies on the following theorem due to Coppersmith [2], which is proved using Lattice Basis Reduction techniques.

Theorem 1. ***(Coppersmith)*** *Let $N = pq$ denote an RSA modulus of length n bits. In polynomial time we can factor N if we know the $n/4$ LS bits of p.*

The BDF attack takes as input the public modulus N of length n bit, the low public exponent e, and an integer d_0 of length $n/4$ bits, consisting of the $n/4$ LS bits of the secret exponent d, i.e. $d_0 = d \pmod{2^{n/4}}$. It then 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 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 by Theorem 1 (since the algorithm terminates with failure in polynomial time even when $x_i \neq \{p_0, q_0\} \pmod{2^{n/4}}$), the attack factors N within time bound $|X| \cdot T_{Cop}(n)$, where $|X|$ denotes the cardinality of X and $T_{Cop}(n)$ is the polynomial running time bound for Coppersmith's algorithm.

The central part of the attack is the construction of the set X since it must have a cardinality 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. The modular key generation equation $ed = 1 \bmod \phi(N)$ implies the integer equation $ed = 1 + k\phi(N)$ for some unique positive integer k. Since the function $f(x) = N + 1 - (x + N/x)$ evaluates to $\phi(N)$ at $x = p$ and $x = q$, it follows that p and q are roots of the quadratic equation $(ed-1) \cdot x - k \cdot xf(x) = 0$. Thus, using the fact that $d_0 = d \bmod 2^{n/4}$ is known, we see that $p_0 = p \bmod 2^{n/4}$ and $q_0 = q \bmod 2^{n/4}$ are roots of the modular equation:

$$kx^2 + (ed_0 - 1 - k(N+1))x + kN = 0 \pmod{2^{n/4}} \quad (1)$$

All the parameters defining (1) can be computed by the attacker with the exception of k. However, assuming that both e and d are smaller than $\phi(N)$, it is easy to see that $k \in \{1, 2, ..., e-1\}$. Since e is small, this set can be searched. So the set X of candidates is generated as follows: for each candidate $k' \in$

$\{1, ..., e-1\}$ for the true value of k, the attacker computes (in time polynomial in n, by lifting solutions modulo 2 to solutions modulo higher powers of 2 until the modulus $2^{n/4}$ is reached) up to $S(k')$ solutions of (1) with the unknown k replaced by k'. Here $S(k')$ is an upper bound on the number of solutions to (1) expected if k was equal to k'. Thus the cardinality $|X| = \sum_{k'=1}^{e-1} S(k')$. The number of solutions $S(k')$ is found by noting that the linear coefficient of (1) is equal to $(k-k')\phi(N) - k'(p+q)$ modulo $2^{n/4}$, which for $k = k'$ reduces to $-k'(p+q)$, so $S(k')$ is the number of solutions to:

$$k'(x^2 - (p+q)x + N) = 0 \pmod{2^{n/4}} \tag{2}$$

Dividing (2) by $m_2(k')$ (where $m_2(z)$ denotes the 2-multiplicity of z) and multiplying by the multiplicative inverse of $odd(k') = k'/2^{m_2(k')}$, we find that $S(k') = 2^{m_2(k')} T(m_2(k'))$, where $T(m_2(k'))$ is the number of solutions to:

$$x^2 - (p+q)x + pq = 0 \pmod{2^{n/4-m_2(k')}}. \tag{3}$$

Thus we have:

$$|X| = \sum_{k'=1}^{e-1} 2^{m_2(k')} \cdot T(m_2(k')) \tag{4}$$

In their paper [1], BDF make the following incorrect deduction:

$$T(m_2(k')) \leq 2 \text{ for all } k' \in \{1, ..., e-1\} \tag{5}$$

It is the estimate (5) that we wish to correct in this paper. Putting (5) in (4) leads to the conclusion that

$$|X| < 2 \cdot \sum_{m=0}^{\lfloor \log_2 e \rfloor} 2^m \cdot (\sum_{k' \in H(m)} 1) < 2e\lfloor \log_2 e \rfloor, \tag{6}$$

where the set $H(m) \stackrel{\text{def}}{=} \{k' \in \{1, ..., e-1\} : m_2(k') = m\}$. This gives a total running time bound $2e\lfloor \log_2 e \rfloor \cdot T_{Cop}(n)$ which is necessarily polynomial in n since e is small.

We remark here that the above description differs slightly from that presented by BDF to fix an independent minor problem of the analysis presented by BDF. In particular, BDF used the same symbol to represent both the true value k which is hidden and fixed 'inside' $ed_0 = 1 + k\phi(N) \bmod 2^{n/4}$, and the trial k' which is swept in the set $\{1, ..., e-1\}$, and hence were led to the incorrect claim that the number of solutions to (1) with k replaced by any $k' \in \{1, ..., e-1\}$ is the same as that when $k = k'$, namely $S(k')$ using the above notation. We fix this without affecting the analysis by making the attacker reject a value of k' as clearly not equal to k if for this value (1) has more than $S(k')$ solutions (while BDF suggested to try *all* solutions for each k', which would require a separate proof that this number is not greater than $S(k')$).

3 A Lemma

We first present a lemma on taking square-roots modulo 2^γ, which will be useful in the next two sections.

Lemma 1. *The set of solutions to the modular equation* $x^2 = c \pmod{2^\gamma}$ *is summarised as follows. Let* $m = m_2(c)$ *and* $d = odd(c)$.

(i) For the case $\gamma \leq m$, *there are* $2^{\lfloor \gamma/2 \rfloor}$ *solutions of the form* $x = r \cdot 2^{\lceil \gamma/2 \rceil} \pmod{2^\gamma}$ *for* $r \in \{0, ..., 2^{\lfloor \gamma/2 \rfloor} - 1\}$.

(ii) For the case $\gamma > m$, *there are no solutions if* m *is odd. Otherwise, if* m *is even, there are three subcases.*

For $\gamma = m + 1$ *there are* $2^{m/2}$ *solutions.*

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

For $\gamma \geq m + 3$, *there are* $4 \cdot 2^{m/2}$ *solutions if* $d = 1 \pmod 8$ *and none otherwise.*

These solutions have the form $x = r \cdot 2^{m/2} \pmod{2^\gamma}$, *where* $r = \pm s + \delta \cdot 2^{\gamma - m - 1} + t \cdot 2^{\gamma - m} \pmod{2^{\gamma - m/2}}$, $\delta \in \{0, 1\}$ *(*$\delta = 0$ *when* $\gamma = m + 1$*),* $t \in \{0, ..., 2^{m/2} - 1\}$ *and* s *is any solution to* $s^2 = d \pmod{2^{\gamma - m}}$.

Proof. First we note that the given equation $x^2 = c \pmod{2^\gamma}$ is equivalent to

$$m_2(x^2 - c) \geq \gamma, \tag{7}$$

where $m_2(z)$ denotes the 2-multiplicity of z. For the case $\gamma \leq m$, we have $c = 0 \pmod{2^\gamma}$, so $x^2 = 0 \pmod{2^\gamma}$ which is equivalent to $m_2(x^2) = 2m_2(x) \geq \gamma$, or $m_2(x) \geq \lceil \gamma/2 \rceil$, as stated. For the case $\gamma > m$, it can be verified that (7) is equivalent to the conditions (i) $m_2(x^2) = m$ and (ii) $m_2(r^2 - d) \geq \gamma - m$, where $r \stackrel{\text{def}}{=} odd(x)$. From (i) we have that m is even and $x = r \cdot 2^{m/2} \pmod{2^\gamma}$ for odd r, and (ii) has the equivalent form (iii) $r^2 = d \pmod{2^{\gamma - m}}$. Each distinct solution r_0 to (iii) modulo $2^{\gamma - m}$ gives rise to exactly $2^{m/2}$ distinct solutions of (7) modulo 2^γ of the form $r_0 2^{m/2} + l \cdot 2^{\gamma - m/2}$ for any $l \in \{0, ..., 2^{m/2} - 1\}$. For $\gamma - m = 1$ and $\gamma - m = 2$ one can check that (iii) has the only solutions $r = 1 \pmod 2$ and $r = \pm 1 \pmod 4$ respectively, and no solutions in the latter case if $d = 3 \pmod 4$. For $\gamma - m \geq 3$, suppose that s is a solution of (iii). Then it is readily verified that $r = \pm s + \delta \cdot 2^{\gamma - m - 1}$ for $\delta \in \{0, 1\}$ are 4 distinct solutions to (iii) modulo $2^{\gamma - m}$. The lack of any additional solutions and the existence of a solution s is shown in ([10], pages 182-184). One can check that for $\gamma - m = 3$, (iii) has no solutions if $d \neq 1 \pmod 8$, from which the stated result follows for $\gamma - m \geq 3$. This completes the proof of the lemma. □

4 Correction to BDF Attack Time Estimate

We now give the correct estimate for the number of solutions $T(m_2(k'))$ to (3). In their analysis, BDF state correctly that (3) has at most 2 solutions modulo 2 (in fact it has exactly 1 such solution $x = 1 \bmod 2$), but then suggest the use of

Hensel lifting to show that this number of solutions is preserved modulo arbitrary high powers of 2, including in particular $2^{n/4-m_2(k')}$. However, for a polynomial $f(.)$, Hensel's lemma(see [8]) applies only for lifting *non-singular* solutions x of $f(x) = 0 \bmod 2$, i.e. those for which $f'(x) \neq 0 \bmod 2$, where $f'(.)$ denotes the derivative of $f(.)$. But in the case of (3), all solutions are singular.

Theorem 2. *Define* $t_{k'} \stackrel{\text{def}}{=} m_2(k')$ *and* $t_{p-q} \stackrel{\text{def}}{=} m_2(p-q)$. *The number of solutions to Equation (3) is given by*

$$T(t_{k'}) = \begin{cases} 2^{t_{p-q}+\upsilon} & \text{if } t_{k'} < n/4 - 2(t_{p-q}-1) \\ 2^{\lfloor (n/4-t_{k'})/2 \rfloor} & \text{if } t_{k'} \geq n/4 - 2(t_{p-q}-1) \end{cases} \tag{8}$$

where $\upsilon = -1$ *when* $t(k') = n/4 - 2(t_{p-q}-1) - 1$, $\upsilon = 0$ *when* $t(k') = n/4 - 2(t_{p-q}-1) - 2$, *and* $\upsilon = +1$ *when* $t(k') \leq n/4 - 2(t_{p-q}-1) - 3$.

Let $\eta = n/4 - t_{k'}$. *In the case* $t_{k'} < n/4 - 2(t_{p-q}-1)$, *the solutions have the form* $x = (\{p,q\} \bmod 2^{\eta - t_{p-q}}) + r' \cdot 2^{\eta - t_{p-q}} \pmod{2^\eta}$. *In the case* $t_{k'} \geq n/4 - 2(t_{p-q}-1)$, *the solutions have the form* $x = (p \bmod 2^{\lfloor \eta/2 \rfloor}) + r' \cdot 2^{\lceil \eta/2 \rceil} \pmod{2^\eta}$.

Proof. By 'completing the square', since $p+q$ is even, we can write (3) in the equivalent form $(x - (p+q)/2)^2 = 1/4((p+q)^2 - 4pq) = ((p-q)/2)^2 \bmod 2^\eta$. Applying Lemma 1 with $\gamma = n/4 - t_{k'}$ and $c = ((p-q)/2)^2$, the claimed number of solutions follows immediately. Writing $p = l + p_H \cdot 2^{t_{p-q}}$ and $q = l + q_H \cdot 2^{t_{p-q}}$, where exactly one of p_H and q_H is odd (so that $l < 2^{t_{p-q}}$ represents the t_{p-q} shared LS bits of p and q) we have that $(p+q)/2 = l + (p_H + q_H) \cdot 2^{t_{p-q}-1}$ and $(p-q)/2 = (p_H - q_H) \cdot 2^{t_{p-q}-1}$. From Lemma 1, the solutions in the case $t_{k'} \geq n/4 - 2(t_{p-q}-1)$ are $x = (p+q)/2 + r \cdot 2^{\lceil \eta/2 \rceil} \pmod{2^\eta}$ and since r is arbitrary, we have $x = (p+q)/2 \bmod 2^{\lceil \eta/2 \rceil} + r' \cdot 2^{\lceil \eta/2 \rceil} \pmod{2^\eta}$ for arbitrary r', which gives the stated result $x = l + r' \cdot 2^{\lceil \eta/2 \rceil} \pmod{2^\eta}$ since $\lceil \eta/2 \rceil \leq t_{p-q}$ and $(p_H + q_H) 2^{(t_{p-q}-1)} = 0 \pmod{2^{t_{p-q}}}$. Similarly, for the case $t_{k'} < n/4 - 2(t_{p-q}-1)$, we apply Lemma 1 with the solution $s = odd(p-q) \pmod{2^{\eta - 2(t_{p-q}-1)}}$ to $s^2 = odd((p-q)^2/4) \pmod{2^{\eta - 2(t_{p-q}-1)}}$, giving $x = l + (p_H + q_H) 2^{t_{p-q}-1} + (\pm(p_H + q_H) + \delta \cdot 2^{\eta - 2(t_{p-q}-1) - 1} + t \cdot 2^{\eta - 2(t_{p-q}-1)}) \cdot 2^{t_{p-q}-1} \pmod{2^\eta}$, which simplifies to the desired result $x = l + (\{p_H, q_H\}) \cdot 2^{t_{p-q}} + (2t + \delta) 2^{n/4 - t_{p-q}} \pmod{2^\eta}$. □

With this result, we see that the BDF attack becomes intractable for $(m - LSbS)$ moduli with sufficiently large $m = t_{p-q}$. Specifically, the running time bound stated by BDF must be increased, and we can state the following corrected running time estimate for the BDF attack.

Corollary 1. *Given the* $(n/4)$ *LS bits of* d, *the BDF attack factors* N *within the following time bound:*

$$T_{BDF}(n) \leq \begin{cases} 2e \lfloor \log_2 e \rfloor 2^{t_{p-q}+1} T_{Cop}(n) & \text{if } 2(t_{p-q}-1) < n/4 \\ 2e \lfloor \log_2 e \rfloor 2^{n/8} T_{Cop}(n) & \text{if } 2(t_{p-q}-1) \geq n/4 \end{cases} \tag{9}$$

Proof. Recall that $T_{BDF}(n) \leq |X| \cdot T_{Cop}(n)$. From (4), we have the following bound on the cardinality of the set X constructed by the BDF attack:

$$|X| = \sum_{m=0}^{\lfloor log_2 e \rfloor} 2^m T(m) \lceil \lfloor (e-1)/2^m \rfloor /2 \rceil \tag{10}$$

$$< \sum_{m=0}^{\lfloor log_2 e \rfloor} 2^m T(m)(e/2^m + 1)/2 \tag{11}$$

$$< e \cdot \sum_{m=0}^{\lfloor log_2 e \rfloor} T(m). \tag{12}$$

Now from Theorem 2 we see that if $2(t_{p-q} - 1) < n/4$, then, defining $\alpha_1 \stackrel{\text{def}}{=} \min(\lfloor log_2 e \rfloor, n/4 - 2(t_{p-q} - 1) - 1)$, we have:

$$\sum_{m=0}^{\lfloor log_2 e \rfloor} T(m) = \sum_{m=0}^{\alpha_1} 2^{t_{p-q}+1} + \sum_{m=\alpha_1+1}^{\lfloor log_2 e \rfloor} 2^{\lfloor (n/4-m)/2 \rfloor} \tag{13}$$

$$\leq \sum_{m=0}^{\lfloor log_2 e \rfloor} 2^{t_{p-q}+1} < 2\lfloor log_2 e \rfloor 2^{t_{p-q}+1}, \tag{14}$$

where the second term in the right hand side of (13) is defined to be zero for $\alpha_1 = \lfloor log_2 e \rfloor$. From (14), the first case of the corollary follows using (12). Similarly, for the second case $2(t_{p-q} - 1) \geq n/4$ we have

$$\sum_{m=0}^{\lfloor log_2 e \rfloor} T(m) = \sum_{m=0}^{\lfloor log_2 e \rfloor} 2^{\lfloor (n/4-m)/2 \rfloor} \tag{15}$$

$$< \sum_{m=0}^{\lfloor log_2 e \rfloor} 2^{n/8} < 2\lfloor log_2 e \rfloor 2^{n/8} \tag{16}$$

This gives, using (12), the second claim, which completes the proof. □

We note that when the prime factors p and q of N are chosen randomly and independently, one would heuristically expect that $\Pr[t_{p-q} = m] = 1/2^m$ for $m \geq 1$. Therefore in this case t_{p-q} would be small with high probability, so our result does not imply the intractability of the BDF attack in this (most common) case.

The above results can be generalized in a straightforward way to the case when the attacker is given any number $z \geq n/4$ of LS bits of d. In this case, the number of arbitrary MS bits in the solutions to Equation (3) with $n/4$ replaced by z is about $z/2$ when $z \leq 2t_{p-q}$ and about t_{p-q} when $z \geq 2t_{p-q}$. However, since only the $n/4$ LS bits of the correct solutions are needed by Coppersmith's algorithm, only $n/4 - z/2$ bits and $n/4 + t_{p-q} - z$ bits need be searched in the two cases, respectively, and so the power of 2 in the running time estimate for

the attack is about $2^{n/4-z/2}$ for $z \leq 2t_{p-q}$ and about $2^{n/4+t_{p-q}-z}$ for $z \geq 2t_{p-q}$. Therefore, the attack requires about $z = n/4 + t_{p-q}$ LS bits of d in order to be tractable.

5 Properties of (m-LSbS) RSA Moduli

In the previous section we showed that the BDF partial key exposure attack for a low-exponent RSA system using $(m - LSbS)$ RSA moduli is intractable (i.e. requires time exponential in the modulus length n), if m is large, i.e. increases proportionally to n. However, it is natural to ask whether it is possible to modify the BDF attack or find another attack which, given the $n/4$ LS bits of d, factors N in time polynomial in n even for large m. We have not found such an attack when $m \leq n/4(1-\epsilon)$, where ϵ is a positive constant. Such an attack may exist, but the following result shows that finding such an attack for low-exponent RSA systems using $(m - LSbS)$ moduli with $m \geq n/8$ implies finding a poly-time factoring algorithm for these $(m - LSbS)$ moduli.

Theorem 3. *Let (N, e, d) be an RSA key pair, where $N = pq$ is a $(m - LSbS)$ RSA modulus of length n bits. Let $\mathsf{A}(.,.,.)$ denote a partial key exposure attack algorithm that, given up to $2m$ LS bits of d and the public pair (N, e), factors N in time T_A. Then we can construct a factoring algorithm $\mathsf{F}(.,.)$, that given only (N, e), factors N with time bound $O(n) \cdot (e \cdot (T_A + O(n^2)) + O(n^2))$.*

Proof. We show how to construct the factoring algorithm $\mathsf{F}(.,.)$. Given (N, e), F simply computes d_0, the $2m$ LS bits of d, and runs A on input (N, e, d_0). To find d_0, F does the following. First, F guesses the number of shared bits m (at most $n/2$ guesses suffice to find the right value). Then, F solves

$$x^2 = N \pmod{2^m} \tag{17}$$

Writing $p = l + p_H \cdot 2^m$ and $q = l + q_H \cdot 2^m$ with $l < 2^m$ representing the m shared LS bits of p and q, we see that $N = pq = l^2 \pmod{2^m}$. Applying Lemma 1, the equation (17) has 4 solutions modulo 2^m of the form $x = \pm l + \delta 2^{m-1} \pmod{2^m}$ with $\delta \in \{0, 1\}$. Thus l can be guessed correctly after at most 4 trials (we note that (17) can be solved by lifting solutions modulo 2 to higher powers in time $O(m^2)$). Since $N = (l + p_H \cdot 2^m)(l + q_H \cdot 2^m) = l^2 + l(p_H + q_H)2^m + p_H q_H 2^{2m}$ then $l(p_H + q_H) = (N - l^2)/2^m \pmod{2^m}$ and since l is odd, it has a multiplicative inverse l^{-1} modulo 2^m. So F can compute $s_H \stackrel{\text{def}}{=} l^{-1}(N - l^2)/2^m = p_H + q_H \pmod{2^m}$. Therefore F knows $s_0 \stackrel{\text{def}}{=} s_H \cdot 2^m + 2l = p + q \pmod{2^{2m}}$, from which the desired LS bits of d can be computed using $d_0 = e^{-1} \cdot (1 + k(N + 1 - s_0)) \bmod 2^{2m}$, where e^{-1} is the multiplicative inverse of e modulo 2^{2m}, which exists since e is odd. Since it is known that $k \in \{1, ..., e-1\}$, F can guess k correctly using less than e guesses. The computation time per guessed value of k is bounded as $O(n^2) + T_A$ (we assume that the bound T_A is easily computable by F so that it can halt A after time T_A regardless of A's behaviour on inputs with an incorrect trial value for d_0), and the time for all other computations per

guessed value of m are bounded as $O(n^2)$. Hence, since the number of guesses needed to find m is $O(n)$, we conclude that F factors N within the claimed time bound. □

The proof of the above result shows that for low-exponent RSA, $(m - LSbS)$ RSA moduli 'leak' the $2m$ LS bits of d. Therefore, it is easily generalized (by changing A from a factoring attacker to any RSA attacker) to show that under the sole assumption that low-exponent RSA using an $(m - LSbS)$ modulus is secure against conventional poly-time attackers (with no access to secret bits), the $2m$ LS bits of d can be made public without compromising the security of the system against poly-time attackers, who can compute these bits by themselves (within a small set of uncertainty).

If the assumption that low-exponent RSA with $(m - LSbS)$ moduli is secure holds for some m, then as mentioned in Sect. 1, this result can be useful (in conjunction with BDF's result that low-exponent RSA also leaks the $n/2$ MS bits of d) in the construction of a fast SASC protocol for securely speeding up the RSA decryption operation with an $(m - LSbS)$ modulus N. To be specific, we illustrate this for the most common SASC application, namely server-aided signature generation by a card on a message M (in practice M would be an element of $\mathbb{Z}_N^*$, obtained from the real message by one-way hashing). We 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. Define $d_{sec} \stackrel{\text{def}}{=} (1/2^{2m}) \cdot \sum_{i=2m}^{n/2-1} d_i 2^i$ and $d_{pub} \stackrel{\text{def}}{=} d - 2^{2m} d_{sec}$. The SASC protocol for computing the signature $s = M^d \bmod N$ consists of the following steps: (1) The card forwards the quadruple (M, d_{pub}, m, N) to the server. (2) The server computes $\beta_1 \stackrel{\text{def}}{=} M^{d_{pub}} \bmod N$ and $\beta_2 \stackrel{\text{def}}{=} M^{2^{2m}} \bmod N$. (3) The server forwards the pair (β_1, β_2) to the card. (4) The card computes $s \stackrel{\text{def}}{=} \beta_1 \beta_2^{d_{sec}} \bmod N$. (5) The card checks if $s^e = M \bmod N$. If not, the card discards s and terminates with an error. (6) The card outputs the signature s on message M. The length of the exponent in the exponentiation (4) performed by the card is only $(n/2 - 2m)$ bits. By use of CRT, this exponentiation can be sped up by a factor of 2. The verification step (5) is required only in order to provide security against active attacks by servers whose responses deviate from the protocol, and can be omitted in cases when the server is trusted to respond correctly. In any case, step (5) increases the computation for the card by only a small fraction since e is small. The communication required by the card is about $2.5n + 2m$ transmitted bits and $2n$ received bits.

A remaining consideration is the security of low-exponent RSA with $(m - LSbS)$ moduli against conventional attacks, in particular factoring of the modulus. It is clear from the proof of Theorem 3 that $(m - LSbS)$ RSA moduli also leak the m shared LS bits of p and q. Therefore one must be careful in choosing m small enough to prevent the use of a factoring algorithm which makes use of this knowledge. As far as we know, the best such algorithm is that of Coppersmith (see Theorem 1), which shows that $(m - LSbS)$ moduli are easy to factor when $m \geq n/4(1 - \epsilon)$, where $2^{\epsilon \cdot n/4}$ is small enough to exhaustively search for the $\epsilon \cdot n/4$ unknown bits of p or q. In fact, Theorem 3 shows that in the case

$m \geq n/4$ one can simply guess an e relatively prime to $\phi(N)$, compute the $n/2$ LS bits of d using the algorithm F above and then the $n/2$ MS bits of d as shown by BDF, then knowing all of d and e, one has a multiple of $\phi(N)$, so it is easy to factor N using Miller's algorithm (see e.g. [9]). But when ϵ is sufficiently large so that a set of size $2^{\epsilon \cdot n/4}$ is infeasible to search, we know of no algorithm which can efficiently factor $(n/4(1-\epsilon) - LSbS)$ RSA moduli. We emphasize that $(m - LSbS)$ RSA moduli satisfy $p - q = r \cdot 2^m$, which does *not* imply that $|p-q|$ is small, a property which is known to allow easier factorization of N (see [12] and [3]).

In practice, generating $(m - LSbS)$ 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 $q = p \bmod 2^m + 2^m + r \cdot 2^{m+1}$ (with a randomly chosen r) for primality, is asymptotically expected to be as efficient as the 'standard' independent primes generation algorithm, where each candidate is chosen as a random odd integer. This is due to a quantitative version of Dirichlet's Theorem (see [10]), which implies that the density of primes less than a bound x in *any* arithmetic progression $q = a \pmod{z}$ (with $\gcd(a, z) = 1$) converges to $(z/\phi(z)) \cdot (1/\ln x)$. For the case $z = 2^\alpha$, we have $2^\alpha/\phi(2^\alpha) = 2$ for all $\alpha \geq 1$. Therefore, the density of primes converges to $2/\ln x$ for both the standard modulus generation search (where $\alpha = 1$ and $a = 1 \bmod 2$), as well as the $(m - LSbS)$ modulus generation search (where $\alpha = m + 1$ and $a = p + 2^m \bmod 2^{m+1}$).

Finally, we mention that Lenstra [4] 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 $(m - LSbS)$ moduli have a potential speedup advantage by leaking bits of d.

6 Conclusions

We have shown that the Boneh-Durfee-Frankel partial key exposure attack on low public exponent RSA systems becomes intractable for $(m - LSbS)$ RSA moduli having prime factors sharing m LS bits, for sufficiently large m. We then proved that if low exponent RSA with an $(m - LSbS)$ modulus is secure against conventional attacks, then it is also secure against partial key exposure attacks accessing up to $2m$ LS bits of the secret exponent d. This can have applications in fast public-server-aided RSA decryption or signature generation. An important problem left open is to characterize the largest m for which low-exponent RSA with an $(m - LSbS)$ modulus is secure, since this defines the limit on the effectiveness of the technique.

Acknowledgements. The authors would like to thank the anonymous referees for their helpful comments.

References

1. 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. See full paper, available from `http://crypto.stanford.edu/~dabo/pubs`.
2. D. Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. of Cryptology*, 10:233–260, 1997.
3. B. de Weger. Cryptanalysis of RSA with small prime difference. Cryptology ePrint Archive, Report 2000/016, 2000. `http://eprint.iacr.org/`.
4. A. Lenstra. Generating RSA Moduli with a Predetermined Portion. In *ASIACRYPT '98*, volume 1514 of *LNCS*, pages 1–10, Berlin, 1998. Springer-Verlag.
5. 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.
6. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of applied cryptography*. Discrete mathematics and its applications. CRC Press, 1997.
7. 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.
8. I. Niven, H. Zuckerman, and H. Montgomery. *An Introduction to the Theory of Numbers*. John Wiley & Sons, fifth edition, 1991.
9. G. Poupard and J. Stern. Short Proofs of Knowledge for Factoring. In *PKC 2000*, volume 1751 of *LNCS*, pages 147–166, Berlin, 2000. Springer-Verlag.
10. D. Redmond. *Number Theory: an introduction*. Number 201 in Monographs and textbooks in pure and applied mathematics. Marcel Dekker, 1996.
11. 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.
12. R. Silverman. Fast Generation of Random, Strong RSA Primes. *CryptoBytes*, 3(1):9–13, 1997.