Reconstruction and Error Correction of RSA Secret Parameters from the MSB Side

Sarkar Santanu, Gupta Sourav Sen, Maitra Subhamoy

Reconstruction and Error Correction of RSA Secret Parameters from the MSB Side

Santanu Sarkar, Sourav Sen Gupta, and Subhamoy Maitra

Indian Statistical Institute, 203 B T Road, Kolkata 700108, India
sarkar.santanu.bir@gmail.com, souravsg_r@isical.ac.in, subho@isical.ac.in

Abstract. This paper discusses the factorization of the RSA modulus when some 'partial information' about the bits of the RSA secret parameters are known. Heninger and Shacham (Crypto 2009) considered the reconstruction of RSA secret parameters from a few randomly known bits, and Henecka, May and Meurer (Crypto 2010) studied the reconstruction of secret parameters when all the bits are known with some probability of error. Both the approaches attempted reconstruction from the least significant side of the parameters. In this paper we provide two new techniques for reconstruction and error correction that work from the most significant side of the parameters. Our idea uses adaptive filtering using a 'sliding window' technique. We provide theoretical as well as experimental results that are competitive with the existing works.

Keywords: Error correction, Prime reconstruction, Random known bits, RSA.

1 Introduction

RSA has been the most popular public key cryptosystem since its inception in 1978. The RSA modulus $N = pq$ is constructed as a product of two large (usually 512 or 1024 bit) primes, and the Euler totient function $\phi(N)$ is used to obtain the exponents e (coprime to $\phi(N)$) and $d = e^{-1} \bmod \phi(N)$. Encryption of plaintext M follows the rule $C = M^e \bmod N$ while the ciphertext C is decrypted as $M = C^d \bmod N$. One can find a complete description of the RSA scheme in [10]. To accelerate the decryption process of RSA, one can use CRT-RSA [9], a variant of the original scheme that uses Chinese Remainder Theorem (CRT). In CRT-RSA, one uses $d_p = d \bmod p - 1$ and $d_q = d \bmod q - 1$, instead of d, for the decryption process. This is the most widely used variant of RSA in practice, and decryption becomes more efficient if one pre-calculates the value of $q^{-1} \bmod p$. Thus, in PKCS [8] standard for the RSA cryptosystem, it is recommended to store the secret parameters as a tuple $SK = (p, q, d, d_p, d_q, q^{-1} \bmod p)$, which we shall henceforth refer to as the *secret key*.

RSA and CRT-RSA have been put through extensive cryptanalysis over the last three decades. An important model of cryptanalysis is the side channel attack, such as fault attacks, timing attacks, power analysis etc., by which an adversary may obtain some information about the secret key SK.

In this paper, we concentrate on the partial information about the secret key SK retrieved using side channel attacks, and how this information may be exploited to mount an attack on RSA. This type of attacks on RSA or CRT-RSA is popularly known as 'partial key exposure' attacks. A majority of the partial key exposure attacks on RSA [1–3, 11] consider the knowledge of contiguous blocks of bits in the secret parameters. While the knowledge of contiguous bits may be hard to obtain, one may gain the knowledge of random bits of the RSA secret parameters via some practical side channel attacks; namely the Cold-Boot attack introduced by Halderman et al. [4].

In Crypto 2009, the Cold-Boot attack was first exploited by Heninger and Shacham [6], who proved that the attacker can factor N in time $\text{poly}(e, \log_2 N)$ if he/she has $\delta \geq 0.27$ fraction of random bits of p, q, d, d_p, d_q, or $\delta \geq 0.42$ fraction of random bits of p, q, d, or $\delta \geq 0.57$ fraction of random bits of p, q. In Africacrypt 2010, Maitra et al. [7] proposed a method for reconstructing p, q from the MSB side, when a few blocks of MSBs of the primes are known.

Recently, in Crypto 2010, Henecka et al. [5] studied the case when all the bits of SK were known, but with some probable error. In this case, one may consider that each bit of the secret parameters in SK is flipped with some probability $\gamma \in [0, \frac{1}{2})$. In [5], the authors proved that one can correct the errors in the secret key SK in polynomial time (for small e) when the error rate is $\gamma < 0.237$ when p, q, d, d_p, d_q are known with error, or $\gamma < 0.160$ when p, q, d are known with error, or $\gamma < 0.084$ when p, q are known with error.

Our Contribution. In this paper, we propose an extension of the aforesaid ideas towards reconstruction and error correction of the RSA secret key SK, starting from the most significant bit (MSB) side, where some partial information about the parameters is known. We present two new algorithms:

- Algorithm 1 (RecSK): Reconstructs RSA secret parameters from the MSB side when random bits of the secret key $(p, q, d, d_p, d_q, q^{-1} \bmod p)$ are known.
- Algorithm 2 (CorSK): Corrects the bit-errors in RSA secret key, starting from the MSB side of the parameters, when all the bits of the secret key $(p, q, d, d_p, d_q, q^{-1} \bmod p)$ are known, but with some probability of error.

The algorithms are described in Section 2 and Section 3 respectively, and we also present relevant experimental results to substantiate our claim.

2 Reconstruction of RSA Secret Key

In this section we propose the first algorithm RecSK which recovers the RSA secret key SK from its partial knowledge. We assume that some random bits of SK are known through some side channel attack on the system. We present a probabilistic polynomial time algorithm whose inputs are these random known bits of the secret parameters of RSA, and the output is the top half of one of the RSA primes (p, say). Our goal is to recover the top $\frac{1}{2}\log_2 p$ many MSBs of prime p and then use the lattice based approach of Coppersmith [2] to factor N.

It is clear if the attacker knows any one of $\{p, q, d, d_p, d_q\}$, he/she can easily factor N. Note that we have four RSA equations connecting these five variables:

$$N = pq, \quad ed = 1 + k\phi(N), \quad ed_p = 1 + k_p(p-1), \quad ed_q = 1 + k_q(q-1) \quad (1)$$

Now when e is small (within the range of a brute force search), one can capture k, k_p, k_q easily as each of these is smaller than e. Let us write $d = d_1 + d_2$, where $d_2 = d \bmod 2^{(\log_2 N)/2}$ and $d_1 = d - d_2$. It is also well known that when e is small, one can find around half of the MSBs of d, that is the portion d_1 (see [1] for details). Hence in Equation (1), we are left with five unknowns p, q, d_p, d_q and d_2. In the next subsection, we propose our algorithm to find these unknown parameters from the knowledge of some random bits.

2.1 The Idea for Reconstruction

Consider the case when few random bits are known from the upper halves of p, q, d_2, d_p, d_q, and assume that we know the first a many contiguous MSBs of p. According to the algorithm proposed in [7], one can recover the first $a - t$ many contiguous MSBs of q with probability greater than $1 - \frac{1}{2^t}$.

Our situation is, however, different than the one considered in [7]. Here, we only know a few random bits of the RSA parameters from the MSB side, not contiguous blocks. That is, in the parameters p, q, d, d_p, d_q, we know about δ fraction of the bits at random (where $0 \leq \delta \leq 1$). We further assume that the known bits of the secret parameters are distributed uniformly at random, i.e., we know δa many bits in each block of size a, and $a(1-\delta)$ bits are unknown.

Our idea for reconstruction provides a two-part solution to this problem. We perform two steps iteratively over the bits of prime p, starting at the MSB:

- Guess the missing bits in a block (of size a, say) to obtain all possibilities.
- Filter using the known bits of q, d_2, d_p, d_q to recover the correct option.

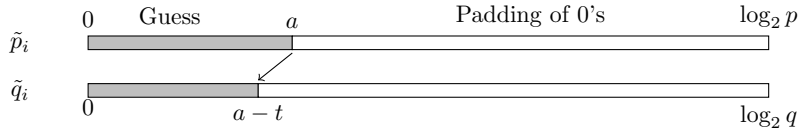
In the Guess routine, we consider all options for the unknown $a(1-\delta)$ many bits in a specific block of p, and construct $2^{a(1-\delta)}$ possibilities for the block. Using each of these options, we mimic the reconstruction idea of [7] to find the first $(a-t)$ many MSBs of q, d_p, d_q, d_2. In the Filter stage, we utilize the bits we know from q, d_p, d_q, d_2 to discard obvious wrong options. We first illustrate our methodology for p and q, and later extend this to utilize all RSA parameters.

2.2 Reconstruction using p and q

Suppose that we know δ fraction of the bits of p and q each, distributed uniformly at random, as discussed. Now, the reconstruction algorithm is as follows.

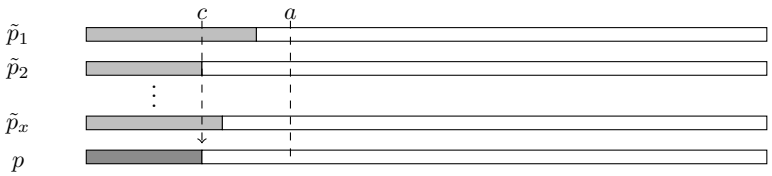
Step 0. In the Guess routine, we generate $2^{a(1-\delta)}$ options for the first a MSBs of p, pad the remaining by 0's, and store in an array A, say. The Filter algorithm is performed on $A = \{\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_k\}$ where $k = |A| = 2^{a(1-\delta)}$.

Step 1. For each option $\tilde{p}_i \in A$, we reconstruct the first $(a-t)$ MSBs of q using the idea of [7], i.e., $\tilde{q}_i = \lfloor \frac{N}{\tilde{p}_i} \rfloor$. Store these options in an array B. We expect the first block of $a-t$ MSBs of $\tilde{q}_i$ to correctly correspond to $\tilde{p}_i$ with probability $1 - \frac{1}{2^t}$ in each of these cases (t is the offset).

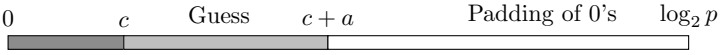


Step 2. Match the known bits of q from this block (first $(a-t)$ MSBs of q) with the corresponding ones in each of these reconstructions $\tilde{q}_i$. If for some $1 \leq l \leq a-t$, the bit $q[l]$ is known but $q[l] \neq \tilde{q}_i[l]$, then there is a mismatch, and we discard $\tilde{q}_i$ from B, and hence $\tilde{p}_i$ from A. If all the known bits of q match with those of $\tilde{q}_i$, then only we retain $\tilde{p}_i$ in A. After this match-and-discard stage, the number of remaining options in $A = \{\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_x\}$ is x, which we hope is considerably lower than the initial number of options $k = 2^{a(1-\delta)}$.

Step 3. Each remaining option has some correctly recovered block of MSBs. We try to find the initial contiguous *common portion* out of the x options, i.e., we find the maximum value of c such that c many initial MSBs of all the options are same. If there is only a single option left in A (i.e., $x = 1$), we evidently get $c = a$. As we have chosen only the common bits of all possibilities, we can be certain that these c bits of p are correctly recovered.



Iterate. Now, we take the next block of a bits of p starting at the $(c+1)$-th MSB, and repeat the Guess and Filter routines using the first $(c+a)$ MSBs of p.



The last two steps above (Step 3 and Iterate) constitute the 'sliding window' technique that filters adaptively over the length of p. This process is continued until half of p is recovered from the MSB side. We formalize the efficiency of the reconstruction algorithm in the next section.

2.3 Efficiency of Reconstruction using p and q

In this section, we find the fraction δ of the bits of the primes p and q that is required to be known for our reconstruction algorithm to recover the top half of p successfully. Before we prove the main result, we require the following.

Lemma 1. *Suppose that in a contiguous block of a many bits of p, we know ν fraction of the bits. Also suppose that the first unknown bit of p in this block occurs at the y-th position, starting from the MSB side. Then, the expected value of y is given by $E(y) = \frac{1}{1-\nu} - a\nu^a - \frac{\nu^a}{1-\nu}$.*

Proof. Each bit of p in the block is known with probability ν. The range of values that y can assume within this block is $y \in [1, a]$, and the probability distribution of y follows $\Pr(y = k) = \nu^{k-1}(1-\nu)$. Thus, we obtain the expectation as $E(y) = \sum_{k=1}^{a} k \cdot \Pr(y = k) = (1-\nu)\sum_{k=1}^{a} k \cdot \nu^{k-1} = \frac{1}{1-\nu} - a\nu^a - \frac{\nu^a}{1-\nu}$. □

Note that the expectation can be approximated as $\overline{y} = E(y) \approx \frac{1}{1-\nu}$ when a is large. Now we can state and prove the main technical result of this section.

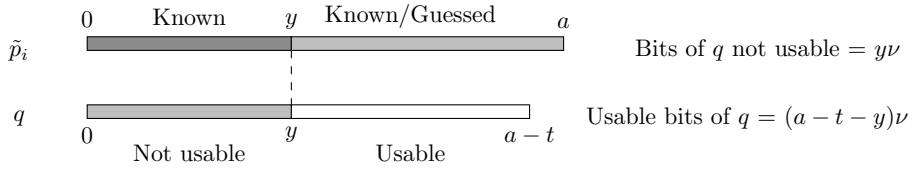
Theorem 1. *Suppose that in each block of size a in the primes p and q, about ν fraction of bits are known. In this case, the reconstruction algorithm described above can successfully recover the top half of p, with probability $\left(1 - \frac{1}{2^t}\right)^{\frac{\log p}{2(a-t)}}$, if*

$$\nu \geq \frac{3}{4} - \frac{1}{4(a-t)}\left(1 + \sqrt{(a-t)^2 - 6(a-t) + 1}\right),$$

where block size a and offset t are used as the parameters for the algorithm.

Proof. In a block of size a in prime p, we know ν fraction of the bits. Thus, there are $a(1-\nu)$ many unknown bits in the block. In Step 0 (Guess routine) of the reconstruction algorithm, this will result in $2^{a(1-\nu)}$ options, i.e., $k = |A| = 2^{a(1-\nu)}$. Each of these options are used to find an approximation for a block of $a-t$ bits of q, following the idea in Step 1. This introduces the probability $1 - \frac{1}{2^t}$ in the result for each block of size $(a-t)$, and there are $\frac{\log p}{2(a-t)}$ many such blocks in our target range, i.e., the top half of p.

In the Filter phase of the algorithm, Step 2 performs the match-and-discard operation using the known bits of this corresponding block of q. Now, we know $(a-t)\nu$ many bits in this block of q, but not all of the bits will be useful in filtering the options. Suppose that the first unknown bit of p in the block of size a occurs at a position y, starting from the MSB. Then, the known bits of q within the range $[1, y]$ in the block of size $(a-t)$ under consideration will have no effect on the unknown bits of p, and hence we will not be able to use about $y\nu$ bits of q for reconstruction.



It is expected that we will be able to use about $(a - t - \overline{y})\nu$ many bits of q in the Filter routine, where the mean value $\overline{y} \approx \frac{1}{1-\nu}$ is as estimated in Lemma 1. Now, we use the following assumption for filtering through the options of p.

Assumption 1: Each useful bit of q is capable of correcting 1 bit of p, and hence reduces the number of options of p by a factor of 2.

The assumption allows us to estimate the number of remaining options after filtering is done. As we know $(a-t-\overline{y})\nu$ many bits of q, the number of options of p reduces by a factor of $2^{(a-t-\overline{y})\nu}$. Thus, we are left with number of options

$$x = \frac{\text{Initial options } k}{\text{Reduction using Filter}} = \frac{2^{a(1-\nu)}}{2^{(a-t-\overline{y})\nu}} = 2^{a(1-\nu)-(a-t-\overline{y})\nu}$$

Note that we have used a block of $a-t$ bits of q for filtering through a block of a bits of p. Thus, it is expected that we will be left with a block of t bits in p which is not corrected (the last t bits, to be specific). Within these t bits, the number of unknown bits of p is $t(1-\nu)$, and hence the best the Filter routine can do is to keep around $2^{t(1-\nu)}$ remaining options after the match-and-discard stage (Step 2). Thus, the desired outcome of Step 2 is $x \approx 2^{a(1-\nu)-(a-t-\overline{y})\nu} \approx 2^{t(1-\nu)}$, i.e., $a(1-\nu)-(a-t-\overline{y})\nu \approx t(1-\nu)$. Putting the estimated value of $\overline{y} \approx \frac{1}{1-\nu}$, as in Lemma 1, and simplifying the equation, we obtain the condition to be

$$\nu^2 - \left(\frac{3}{2} - \frac{1}{2(a-t)}\right)\nu + \frac{1}{2} \approx 0$$

Solving the quadratic equation in ν, we obtain the desired expression. □

Example 1. Suppose that we take $a = 30$ and $t = 5$ for the reconstruction, where p and q are of bitsize 512 each. From Theorem 1, we get $\nu \geq 52.18\%$, which denotes the fraction of bits to be known from each block of size a in primes p, q. But our goal is to estimate δ, the overall fraction of bits to be known for p, q, such that 52.18% of bits are known in each block with a high probability.

If we know each bit of p or q with probability δ, then the probability that we know ν or more fraction of bits in each block of size a is given by $\sum_{k=a\nu}^{a} \binom{a}{k}\delta^k(1-\delta)^{a-k}$. In our case with $a = 30$ and $\nu \approx 52.18\%$, we observe that $\delta \approx 64\%$ ensures a probability greater than 0.9. Thus for $\delta = 64\%$, our reconstruction algorithm will recover the top half of p with approximately 90% chance of success. □

Experimental Data. We have implemented the code in Sage 4.1 (Linux Ubuntu 8.04) on a machine with Dual Core Intel Pentium D CPU (1.83 GHz, 2 GB RAM, 2 MB Cache). In these experiments, we have taken 1024 bit N and $e = 2^{16} + 1$, as is the case in most practical situations. In all the experiments, we chose $t = 5$.

Blocksize	Offset	Known fraction of bits δ (%)		Time (seconds)
a	t	Theoretical	Experimental	T
25	5	67	67	31
30	5	64	65	92
35	5	63	61	3709

Table 1. Experimental data for reconstruction using random known bits of p and q.

2.4 The General Reconstruction Algorithm

In this section, we extend our idea of reconstruction to all the RSA secret parameters p, q, d, d_p, d_q (except for $q^{-1} \bmod p$). Given δ fraction of bits for each parameter, the following algorithm attempts to reconstruct the top half of p.

Input: N, e, k, k_p, k_q, and the known bits of p, q, d_2, d_p, d_q.
Output: Top half of the MSBs of p.

Choose parameters $a, t, b = 0, c_{old} = 1$;
while $b < \frac{l_N}{2}$ **do**
  For first $(a + b)$ MSBs of p, choose all possible options $\tilde{p}$ and store in an array A;
  $A = \mathsf{Filter}(N, q, d_2, d_p, d_q, A, a, t, b)$;
  If the array A remains unaltered, increase a and go to Step 3;
  if $|A| = 1$ **then**
    Set first $(a + b)$ MSBs of $\tilde{p} = A[1]$ as those of prime p;
    Set $b = a + b$ and $c_{old} = a + b$;
  end
  else
    Find $c \leq a + b$ such that the first c MSBs of all members of A are same;
    If c remains unaltered ($c = c_{old}$), increase a and go to Step 3;
    Set $b = c$ and $c_{old} = c$;
  end
end

Algorithm 1: RecSK: MSB reconstruction algorithm using p, q, d_2, d_p, d_q.

Input: N, the known bits of q, d_2, d_p, d_q, array A and parameters a, t, b.
Output: Modified array A of bit strings.

for $i = 1$ *to* $|A|$ **do**
  $\tilde{p} = A[i]$;
  Calculate $\tilde{q} = \lfloor \frac{N}{\tilde{p}} \rfloor$, $\tilde{d_2} = \lfloor \frac{k(N-\tilde{p}-\tilde{q})}{e} \rfloor$, $\tilde{d_p} = \lfloor \frac{k_p \tilde{p}}{e} \rfloor$, $\tilde{d_q} = \lfloor \frac{k_q \tilde{q}}{e} \rfloor$;
  Compare the known bits from the first $(a + b - t)$ MSBs of q with $\tilde{q}$, d_2 with $\tilde{d_2}$, d_p with $\tilde{d_p}$, and d_q with $\tilde{d_q}$ respectively;
  If any of the above comparisons results in a mismatch, discard corresponding $\tilde{p}$ from A;
end
Return A;

Algorithm 2: Subroutine Filter used in RecSK algorithm.

The efficiency analysis for RecSK extends the analysis performed for primes, and we will obtain a generalized version of Theorem 1, using the known bits of

all RSA secret parameters in SK. Due to space constraints, we do not present the analysis here, but it will be a part of the extended full version of the paper.

However, we support our claim with the help of a few practical reconstructions, as presented in Table 2. In the full version of the paper, we will include the theoretical bounds on δ and more experimental results.

Random bits known from p, q and d			
Blocksize a	Offset t	Known fraction of bits δ (%)	Time T (seconds)
25	5	53	1517
30	5	46	10376

Random bits known from p, q, d, d_p and d_q			
Blocksize a	Offset t	Known fraction of bits δ (%)	Time T (seconds)
25	5	50	1042
30	5	41	15685
35	5	38	259210

Table 2. Experimental data for reconstruction algorithm RecSK.

2.5 Comparison with Existing Work

In the existing literature of RSA secret key reconstruction, we can compare our work to those of Heninger and Shacham [6] and Maitra et al. [7].

Heninger and Shacham [6] were the first to propose the problem and solve it, starting from the LSB side of the RSA parameters. We provide the first solution to this problem starting from the MSB side, and we can compare with the result of [6] as in Table 3. The data clearly shows that our technique yields results of comparable quality to those in [6]. However, the fraction of bits required to be known for the reconstruction in [6] is slightly lower compared to that in case of our method. This is because our algorithm entails a certain probability of success, whereas the approach of [6] is deterministic in nature.

Bits known from	Bits required in [6]	Bits required for RecSK
p, q	57%	61%
p, q, d	42%	46%
p, q, d, d_p, d_q	27%	38%

Table 3. Comparison of our reconstruction algorithm with that of [6].

Maitra et al. [7] were the first to consider the reconstruction of primes p, q from the MSB side, but only when small contiguous blocks of bits are known. The probability of correct reconstruction in our case is same as [7], but we can reconstruct from random bits, which could not be resolved in [7].

3 Error Correction Algorithm

In case of our second problem, instead of knowing a few bits, we consider the situation where all the bits of the secret key SK are known, but there is a probability of error associated with each bit. This is the scenario introduced in [5], where the authors proved that when error rate is less than 0.237, one can correct all errors with expected polynomial time, starting from the LSB side.

We consider the same, but starting with the MSB. Suppose that the upper half of all RSA secret parameters p, q, d_2, d_p, d_q are given, but with some error probability γ associated with each bit. We call the available values of the parameters p', q', d_2', d_p', d_q', each of which is erroneous to some extent. We correct the errors starting from the MSB side of the parameters, and propose the following.

Idea for Error Correction [CorSK]. Choose the parameters a (blocksize) and t (offset) for reconstruction, as before, and also choose a threshold for error to be used in this approach.

Step 0. In the Guess routine, we generate 2^a (all possible) options for the first a MSBs of p, pad the remaining portion by 0's, and store in an array A, say.

Step 1. For each $\tilde{p}_i \in A$, we reconstruct first $(a-t)$ MSBs of other parameters:

$$\tilde{q}_i = \left\lfloor \frac{N}{\tilde{p}_i} \right\rfloor, \quad \tilde{d}_{2i} = \left\lfloor \frac{k(N - \tilde{p}_i - \tilde{q}_i)}{e} \right\rfloor, \quad \tilde{d}_{p_i} = \left\lfloor \frac{k_p \tilde{p}_i}{e} \right\rfloor, \quad \tilde{d}_{q_i} = \left\lfloor \frac{k_q \tilde{q}_i}{e} \right\rfloor$$

Step 2. For each i, find the Hamming distance of $\tilde{p}_i, \tilde{q}_i, \tilde{d}_{2i}, \tilde{d}_{p_i}, \tilde{d}_{q_i}$ with the available values p', q', d_2', d_p', d_q' and calculate the sum of all these Hamming distances. The Hamming distance is calculated only for the concerned block of size a in all the parameters, and the sum is considered to be a measure for error. If this sum is less than the predefined threshold, we retain $\tilde{p}_i$ in A, and otherwise we discard $\tilde{p}_i$ from A to reduce the number of options.

Step 3. Find the maximum value of c such that c many initial MSBs of all the options are same, i.e., $\tilde{p}_1[l] = \tilde{p}_2[l] = \cdots = \tilde{p}_x[l]$ for all $1 \leq l \leq c$ but not for $c < l \leq a$. As we have chosen the common bits of all possibilities, these c bits of p are correctly recovered.

Iterate. We take the next block of a bits of p starting at the $(c+1)$-th MSB, and repeat the Guess and Filter routines using the first $(c+a)$ MSBs of p.

As before, this process is continued until half of p is recovered from the MSB side, after which one can factorize N using the idea of Coppersmith [2]. We will present our error correction algorithm CorSK more formally in the full version of the paper. The efficiency analysis will follow similar lines to those in [5] and the idea of Theorem 1. We will present the complete analysis in the extended full version of the paper. In case of practical experiments with our idea for error correction, we have so far been successful in correcting approximately 14% bit error in the SK parameters, using a block size $a = 14$ and offset $t = 4$. Note that this result is competitive with respect to the results published in [5], which could correct 17% bit error with similar block size. In the full paper, we shall present more experimental results as well as the corresponding theoretical bounds.

4 Conclusion

In this paper, we have discussed the factorization of $N = pq$, where the encryption exponent e is small, and partial information about the bits of the RSA secret parameters $(p, q, d, d_p, d_q, q^{-1} \bmod p)$ is known. We have provided two solutions for reconstruction of RSA secret key from MSB side when random bits are known, and also when all the bits are known with some probability of error. In the final (full) version of the paper, we will include the following as well.

- Complete theoretical analysis of the general reconstruction algorithm RecSK.
- Formal algorithm and analysis of the error correction algorithm CorSK.
- Detailed experimental results to support the algorithms and related results.

We will also provide a lattice based general solution to the case where a large chunk of bits of the RSA parameters are missing from the beginning at the MSB side. Note that in such a case (50 initial bits are missing, say) our solution RecSK does not prove effective anymore. Thus a comprehensive description of this proposed lattice based technique will complete our solution for reconstruction and error correction of RSA parameters from the MSB side.

References

1. D. Boneh, G. Durfee and Y. Frankel. Exposing an RSA Private Key Given a Small Fraction of its Bits. Asiacrypt 1998, LNCS 1514, pp. 25-34, 1998.
2. D. Coppersmith. Small solutions to polynomial equations and low exponent vulnerabilities. Journal of Cryptology, 10(4):223–260, 1997.
3. M. Herrmann and A. May. Solving Linear Equations Modulo Divisors: On Factoring Given Any Bits. Proceedings of Asiacrypt 2008: LNCS 5350, pp. 406–424, 2008.
4. J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum and E. W. Felten. Lest We Remember: Cold Boot Attacks on Encryption Keys. In proceedings of USENIX Security Symposium, pp. 45–60, 2008.
5. W. Henecka, A. May and A. Meurer. Correcting Errors in RSA Private Keys. Crypto 2010, LNCS 6223, pp. 351–369, 2010.
6. N. Heninger and H. Shacham. Reconstructing RSA Private Keys from Random Key Bits. Proceedings of Crypto 2009, LNCS 5677, pp. 1–17, 2009.
7. S. Maitra, S. Sarkar and S. Sen Gupta. Factoring RSA modulus Using Prime Reconstruction from random known bits. Africacrypt 2010, LNCS 6055, pp. 82-99, 2010.
8. PKCS #1 Standard for RSA. Available at `http://www.rsa.com/rsalabs/node.asp?id=2125`.
9. J. -J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. Electronic Letters, volume 18, pages 905–907, 1982.
10. R. L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. Communications of ACM, 21(2):158–164, Feb. 1978.
11. R. L. Rivest and A. Shamir. Efficient Factoring based on Partial Information. Proceedings of Eurocrypt 1985, LNCS 219, pp. 31–34, Apr. 1985.