Error Correction of Partially Exposed RSA Private Keys from MSB Side

Santanu Sarkar[1], Sourav Sen Gupta[2⋆], and Subhamoy Maitra[2]

[1] Chennai Mathematical Institute, Chennai, India
`sarkar.santanu.bir@gmail.com`
[2] Indian Statistical Institute, Kolkata, India
`sg.sourav@gmail.com, subho@isical.ac.in`

Abstract. The most popular public key cryptosystem to date has been RSA, whose security primarily relies on the unfeasibility of factoring the modulus, which is a product of two large primes, and on the secrecy of certain RSA parameters. In 2009, the cold-boot attack by Halderman et al presented an important cryptanalytic model where a portion of the secret parameters may be exposed. In this direction, Heninger and Shacham (Crypto 2009) introduced the problem of reconstructing RSA private keys when few random bits from each are known. Later, Henecka, May and Meurer (Crypto 2010) introduced the problem of error-correction in the RSA private keys when all the bits are known with some probability of error. Their approach attempted error-correction from the least significant side of the parameters. In this paper we provide a novel technique for error-correction that works from the most significant side of the parameters. Representative experimental results are provided to substantiate our claim.

Keywords: cryptanalysis, RSA, cold-boot attack, partial key exposure, private keys, error-correction

1 Introduction

RSA has been the most well known and commercially used public key cryptosystem since its inception in 1978 by Rivest, Shamir and Adleman. One can find a complete description of the RSA scheme in [17]. Let us briefly state the algorithm, as follows.

Let $N = pq$ where p and q are primes. By definition of the Euler totient function, $\phi(N) = (p-1)(q-1)$.

- KeyGen: Choose e co-prime to $\phi(N)$. Find d such that $ed \equiv 1 \bmod \phi(N)$.
- KeyDist: Publish public key $\langle N, e\rangle$ and keep private key $\langle N, d\rangle$ secret.
- Encrypt: For plaintext $M \in \mathbb{Z}_N$, ciphertext $C = M^e \bmod N$.
- Decrypt: For ciphertext C, plaintext $M = C^d \bmod N$.

⋆ Supported by DRDO sponsored project Centre of Excellence in Cryptology (CoEC), under MOC ERIP/ER/1009002/M/01/1319/788/D(R&D) of ER&IPR, DRDO.

To accelerate the decryption process of RSA, one can use CRT-RSA [15], 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. Decryption becomes more efficient if one pre-calculates the value of $q^{-1} \bmod p$. Thus, in PKCS [14] standard for the RSA cryptosystem, it is recommended to store the RSA private keys as a tuple

$$PK = (p, q, d, d_p, d_q, q^{-1} \bmod p),$$

which we shall henceforth refer to as the *private key* of RSA. Note that there are other ways of speeding up RSA by choosing bit patterns in the primes (as a non-exhaustive list, one may refer to [3, 10, 19] and the references therein), but CRT-RSA is the most convenient and widely popular variant in practice.

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 private key PK. In this paper, we concentrate on the partial information about the private key PK 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, based on certain 'partial information' about the private keys, is popularly known as 'partial key exposure' attacks.

1.1 Cold-boot attack on RSA

The cold-boot attack, a new paradigm of partial key exposure side-channel attacks on cryptographic primitives, was introduced by Halderman et al [4] in 2009. It is assumed that the attacker has physical access to the system, and he/she can run a 'cold reboot' to restart the machine without destroying its memory contents. The main idea of the attack thereafter is to exploit the data remanence property of the memory, usually based on DRAM or SRAM technologies, to extract relevant information about the cryptosystem functioning on the machine. In [4], it was experimentally verified that a generic computer memory retains a considerable portion of its data even minutes after the cold-boot; thus allowing the attacked to successfully recover sensitive information.

This technique was exploited in [4] to identify and extract fully-retained AES and RSA keys from the memory, even after a few minutes of the cold-boot. The method was further extended to attack practical disk encryption systems like BitLocker, FileVault, TrueCrypt, dm-crypt and Loop-AES [4]. Detailed information about the motivation and consequences of cold-boot attack is presented by the authors of [4] at `https://citp.princeton.edu/research/memory/`.

Partial key exposure from cold-boot attack: Note that all the attacks in [4] exploit the fact that identification and extraction of fully-retained keys are possible from the computer memory. However, the attacker may not be as lucky; he/she may only get a partial information about the secret keys from the memory. In fact, this is the most practical case to assume, considering that the attacker may retrieve the remanent information from the memory after a certain amount of time, and some parts of the memory may have already decayed. In such a scenario, the cold-boot attack motivates a side channel cryptanalysis on RSA where some bits of PK are revealed but not the entire private key. This provides a natural motivation for partial key exposure attack on RSA.

Previous partial key exposure attacks on RSA: Rivest and Shamir [16] pioneered partial key exposure attacks by factoring the RSA modulus $N = pq$ given a contiguous two-third fraction of the bits from the least significant side of one of the primes p or q. Later, a seminal paper [2] by Coppersmith proved that factorization of the RSA modulus can be achieved given half of the MSBs of a factor. His method used LLL [11] lattice reduction technique to solve for small solutions to modular equations. This method triggered a host of research in the field of lattice based factorization, e.g., the works by Howgrave-Graham [8], Jochemsz and May [9].

Note that these results require knowledge of contiguous blocks of bits of the RSA private keys. However, in an actual practical scenario of cold-boot attacks, it is more likely that an adversary will gain the knowledge of random bits of the RSA parameters instead of contiguous blocks. In this model, the application of the earlier factorization methods prove insufficient, and one requires a way to extract more information out of the random bits obtained via the side channel attacks. In [7], it has been shown how N can be factored with the knowledge of a random subset of the bits (distributed over small contiguous blocks) in one of the primes. But still, that did not address the case of partial random-bit key exposure in case of cold-boot attacks; the following did.

Reconstruction of RSA private key from random bits. In Crypto 2009, the cold-boot partial key exposure attack [4] was exploited for the first time against RSA by Heninger and Shacham [6]. In this work, the random bits of both the primes are considered unlike the earlier works (e.g., [2, 1, 7]) where knowledge of the bits of a single prime have been exploited. The authors proved that the attacker can factor N in time 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
$\delta \geq 0.59$ fraction of random bits of p, q.

The algorithm proposed in [6] exploits a modified brute force search of the unknown bits in the RSA private keys with the help of smart pruning of wrong solution paths in the search tree. This pruning is done using the partial knowledge of the random bits in PK; hence the fraction of required known bits decreases as we increase the number of parameters involved in the pruning process.

Error-correction of RSA private key. In Crypto 2010, Henecka et al [5] studied the case when all the bits of PK were known, but with some probability of error in each bit. One may consider that each bit of the parameters in PK is flipped with some probability $\gamma \in [0, \frac{1}{2})$. In [5], the authors proved that one can correct the errors in key PK in time $\text{poly}(e, \log_2 N)$ 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.

The algorithm proposed in [5] guesses the bits of one of the primes and uses the reconstruction approach of [6] as a subroutine to get approximations of the other parameters in PK. The verification of each guess is done by comparing the Hamming distance of the guess with the erroneous version of PK obtained through side-channel attacks. This is equivalent to pruning the search space towards the correct solution, and hence more bit-error can be corrected if one uses more parameters from PK during the pruning phase. A similar avenue was followed in [13] to recover noisy RSA keys using coding-theoretic techniques.

Reconstruction versus error-correction of RSA secret keys: Note that the cold-boot attack generally allows the attacker to extract remanent data from the memory, but the attacker may never be sure of the authenticity of this data. As the pattern of decay is quite unpredictable across the different parts of the memory, it is more likely for the attacker to obtain a 'noisy' information about the secret key, rather than a fragmented information without any noise. From this practical consideration, one should focus more on error-correction of 'noisy' secret keys of RSA, compared to reconstruction from partially recovered random bits. This issue has been highlighted earlier in [5, 13], and we follow the same principle by paying more attention to error-correction of RSA secret keys, rather than reconstruction of the same.

1.2 Motivation of this paper

Both reconstruction [6] and the error-correction [5] routines start from the least significant side of the PK parameters and moves through the complete length towards the most significant side. This is implicitly based on the assumption that the density of the random bits available for reconstruction is the same throughout the parameters and the error-rate for the bits are uniform over the length of each PK entity.

We already have existing algorithms for factorization having either the least significant half [1] or the most significant half of bits [2]. The existing works [6, 5] consider certain information about the bits through the complete length of the secret parameters. For example, in [6], one requires the knowledge of certain proportion of bits over the complete bit representation of the primes and the target is to recover all the bits. The algorithm of [6] proceeds from the least significant bits of the private keys and build them moving towards the most significant side. In [12], we noted that the process can be terminated just after recovering the least significant half of the primes, as after that we can indeed use the existing strategy [1] to recover the remaining bits. That is the reason we revisit the existing strategy [6] with the motivation of recovering only the least significant half of the primes and in such a case, no information is required from the most significant side of the private keys. Following the work of [6], we revisited this idea in [12].

Given this, a natural question is how the respective algorithms perform when we move from the most significant side and have some information from that portion. In [18], we considered the idea for implementing brute-force reconstruction and error-correction on the keys starting from the most significant side. This, in general, seems to be difficult, as the "carries" from multiplication would seem to interfere. Our strategy of [18] overcomes this difficulty by (i) bounding the length of a carry chain and (ii) adopting an iterative approach where a fraction of bits of the private key are guessed. Once the most significant half of a prime is known, the factorization is immediate following the idea of [2].

1.3 Contribution of this paper

This is a follow-up work of [18], where we present further results towards error-correction of RSA private key PK. While in [18], the reconstruction idea has been studied in algorithmic form and detailed experimental results have been presented, the idea of error-correction from MSB side was introduced only. Detailed algorithmic or experimental results had not been presented in [18].

In this paper, we present the complete algorithm for correcting the bit-errors in RSA private key, starting from the MSB side of the parameters. Suppose that all the bits of the private key PK are known, but with some probability of error γ for each bit. Then using our technique, one can correct the errors in the private key PK efficiently when the error rate is of the order of [5] where the problem has been tackled for LSB side.

2 Background: Reconstruction from MSB side [12, 18]

In this section, we explain the existing ideas of reconstructing partially correct RSA secret keys that are required for understanding our strategy.

2.1 Brief description of [12]

We first refer to [12, Section 4] that explains reconstructing the most significant half of the primes p, q given the knowledge of some blocks of bits. Let us recall at this point that, $N = pq$, and the primes p, q are of the same bitsize. We start with the following.

Definition 1. *Define $X[i]$ to be the i-th most significant bit of X with $X[0]$ being the MSB. Also define X_i to be the partial approximation of X where X_i shares the most significant bits 0 through i with X.*

The idea for reconstruction. If one of the primes, p say, is completely available, one can easily obtain $q = N/p$. Now, if a few MSBs of p are known, then one can get an approximation p' of p. This allows us to construct an approximation $q' = \lceil N/p' \rceil$ of the other prime q as well. The idea is to use the known blocks of bits of the primes in a systematic order to repeat this approximation process until we recover half of one of the primes.

Outline of the idea. Suppose that the MSBs $\{0, \ldots, a\}$ of the prime p are known. This gives an approximation p_a of p, and hence an approximation $q' = \lceil N/p_a \rceil$ of q. Let us consider that q' matches q through MSBs $\{0, \ldots, a-t-1\}$, i.e., $q' = q_{a-t-1}$, with some probability depending on t.

Further, given the MSBs $\{a-t, \ldots, 2a\}$ of q, an improved approximation q_{2a} may be constructed using q_{a-t-1} and these known bits. Again, q_{2a} facilitates the construction of a updated approximation $p' = \lceil N/q_{2a} \rceil$, which may be expected to satisfy $p' = p_{2a-t-1}$ with some probability depending on t.

With the knowledge of MSBs $\{2a-t, \ldots, 3a\}$ of p, it is once again possible to obtain an improved approximation p_{3a} of p. This process of constructing approximations is recursed until one can recover the most significant half of one of the primes. The reconstruction idea is illustrated in Fig. 1.

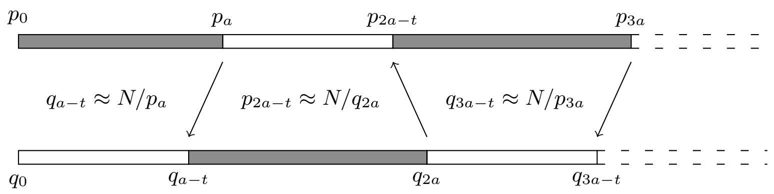


Fig. 1. The feedback mechanism in MSB reconstruction.

The reconstruction algorithm. Let $S = \{0, \ldots, T\}$ denote the set of bit indices from the most significant halves of the primes. Assume that $k = \lfloor T/a \rfloor$ is odd in this case. Consider $U, V \subseteq S$ such that

$$U = \{0, \ldots, a\} \cup \{2a-t, \ldots, 3a\} \cup \cdots \cup \{(k-1)a-t, \ldots, ka\},$$
$$V = \{a-t, \ldots, 2a\} \cup \{3a-t, \ldots, 4a\} \cup \cdots \cup \{ka-t, \ldots, T\}.$$

Further, consider that $p[i]$'s are available for $i \in U$ and $q[j]$'s are also known for $j \in V$. Then, Algorithm 1 can reconstruct T many contiguous most significant bits of the prime q.

The subroutine Correct presented in Algorithm 1 takes as input a partial approximation Y of X and a set of contiguous known bits, $X[i]$ for $i \in \Sigma$, say. It outputs an improved approximation Z of X by correcting the bits of the partial

```
   Input: $N, T$ and $p[i], q[j]$ for all $i \in U$ and $j \in V$
   Output: Contiguous $T$ many MSBs of $q$
1  Initialize: $p_0 := 2^{l_p - 1}$, $q_0 := 2^{l_q - 1}$;
2  $p_a :=$ Correct($p_0$, $p[j]$ for $j \in \{1, \ldots, a\} \subset U$);
3  $q_{a-t} := \lceil \frac{N}{p_a} \rceil$;
4  for $i$ from 2 to $k - 1$ in steps of 2 do
5      $q_{ia} :=$ Correct($q_{(i-1)a-t}$, $q[j]$ for $j \in \{(i-1)a - t, \ldots, ia\} \subset V$);
6      $p_{ia-t-1} := \lceil \frac{N}{q_{ia}} \rceil$;
7      $p_{(i+1)a} :=$ Correct($p_{ia-t-1}$, $p[j]$ for $j \in \{ia - t, \ldots, (i+1)a\} \subset U$);
8      $q_{(i+1)a-t-1} := \lceil \frac{N}{p_{(i+1)a}} \rceil$;
   end
9  $q_T :=$ Correct($q_{ka-t-1}$, $q[j]$ for $j \in \{ka - t, \ldots, T\} \subset V$);
10 Return $q_T$;
```

Algorithm 1: MSB reconstruction algorithm using random blocks of bits from p, q when k is odd [12].

```
   Input: $Y$ and $X[i]$ for $i \in \Sigma$
   Output: $Z$, the correction of $Y$
1  for $j$ from 0 to $l_X$ do
2      if $j \in \Sigma$ then $Z[j] = X[j]$;     // Correct $j$-th MSB if $X[j]$ is known
       else $Z[j] = Y[j]$;                      // Keep $j$-th MSB as $X[j]$ is not known
   end
3  Return $Z$;
```

Algorithm 2: Subroutine Correct used in Algorithm 1 [12].

approximation Y using the knowledge of the known bits. This subroutine works as described is Algorithm 2.

In the case where $k = \lfloor T/a \rfloor$ is even, Algorithm 1 needs to be tweaked a little to work properly. One may use a slightly changed version of $U, V \subseteq S$ such that $U = \{0, \ldots, a\} \cup \{2a - t, \ldots, 3a\} \cup \cdots \cup \{ka - t, \ldots, T\}$ and $V = \{a - t, \ldots, 2a\} \cup \{3a - t, \ldots, 4a\} \cup \cdots \cup \{(k-1)a - t, \ldots, ka\}$. As discussed earlier, $p[i]$'s are known for $i \in U$ and $q[j]$'s are known for $j \in V$.

The work in this section explains in detail the work of [12, Section 4]. The idea is to reconstruct the primes from the MSB side where blocks of bits had to be known. The question remains open whether one can reconstruct the private keys from the MSB side if the random bits are known, but they are not available as contiguous blocks. This problem has been tackled in [18] as we describe below.

2.2 Brief description of [18]

In this section we explain the algorithm RecPK from [18] that recovers the RSA private key PK from its partial knowledge. It is assumed that some random bits of PK are known through some side channel attack on the system. The 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. There are 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. One can 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), there are five unknowns p, q, d_p, d_q and d_2.

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 [12], one can recover the first $(a-t)$ many contiguous MSBs of q with probability greater than $1 - \frac{1}{2^t}$.

The situation is, however, different than the one considered in [12]. 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 $0 \leq \delta \leq 1$). One can further assume that the known bits of the private keys 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.

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

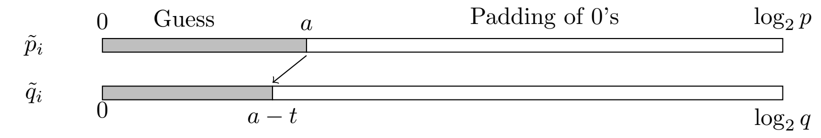
- Guess the missing bits in a block (of size a, say) to obtain all possibilities.
- Filter using the known bits of q, d_2, d_p, d_q to recover the correct option.

In the Guess routine, one considers 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, one can mimic the reconstruction idea of [12] to find the first $(a-t)$ many MSBs of q, d_p, d_q, d_2. In the Filter stage, one can utilize the bits known from q, d_p, d_q, d_2 to discard obvious wrong options.

Reconstruction illustrated 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.

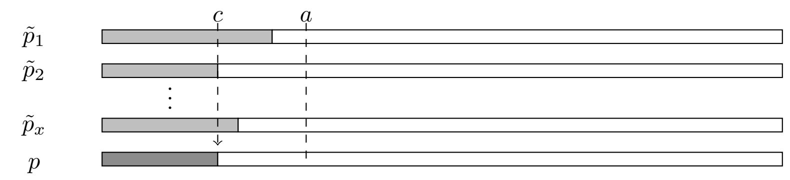
Step 0: In the Guess routine, one generates $2^{a(1-\delta)}$ options for the first a MSBs of p, pads the remaining by 0's, and stores in an array A, say. The Filter algorithm is performed on $A = \{\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_k\}$ where $k = |A| = 2^{a(1-\delta)}$.

Step 1: For each option $\tilde{p}_i \in A$, one reconstructs the first $(a-t)$ MSBs of q using the idea of [12], i.e., $\tilde{q}_i = \lfloor \frac{N}{\tilde{p}_i} \rfloor$. Store these options in an array B. It is expected that the first block of $(a - t)$ MSBs of $\tilde{q}_i$ will correctly correspond to $\tilde{p}_i$ with probability $1 - \frac{1}{2^t}$ in each of these cases (t is the offset).

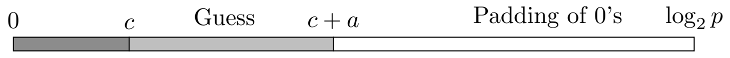


Step 2: Match the known bits of q from this block (first $(a-t)$ MSBs of q) with the corresponding ones in each of these reconstructions $\tilde{q}_i$. If for some $1 \leq l \leq a - t$, the bit $q[l]$ is known but $q[l] \neq \tilde{q}_i[l]$, then there is a mismatch, and one can discard $\tilde{q}_i$ 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 one retains $\tilde{p}_i$ 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 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. One attempts to find the initial contiguous *common portion* out of the x options, i.e., to 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 $c = a$.



Iterate: Now, one can 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.

One may refer to [18] for detailed idea in this regard. We now present how one can modify and refine these ideas for handling the case when each bit may be erroneous with some probability.

3 Error-Correction from MSB side

In case of our second problem, instead of knowing a few bits, we consider the situation where all the bits of the private key PK are known, but there is a probability of error associated with each bit. This scenario was introduced in [5].

The authors proved that when error rate is less than 0.237, one can correct the bit errors from the LSB side with expected polynomial time, if all the parameters p, q, d, d_p and d_q are used in the process.

We consider the same problem of error-correction, but start from the MSB side and correct the error in the top half of the RSA primes as pointed out in [18]. Suppose that the upper halves of all RSA private keys 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 from the MSB side using the following idea.

3.1 Idea for error-correction

Choose the parameters a (blocksize) and t (offset) for reconstruction, and further choose a threshold $T_{a,t,\gamma}$ (depending on the chosen parameters a, t and the error rate γ) for error to be used in this approach.

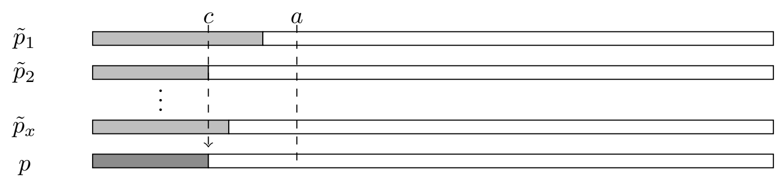
Step 0: In the Guess routine, we generate 2^a (all possible) options for the first a MSBs of p, pad the remaining portion by 0's, and store in an array A, say.

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

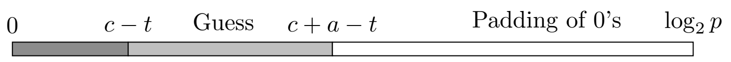
$$\tilde{q}_i = \left\lfloor \frac{N}{\tilde{p}_i} \right\rfloor, \quad \tilde{d}_{2i} = \left\lfloor \frac{k(N - \tilde{p}_i - \tilde{q}_i)}{e} \right\rfloor, \quad \tilde{d}_{p_i} = \left\lfloor \frac{k_p \tilde{p}_i}{e} \right\rfloor, \quad \tilde{d}_{q_i} = \left\lfloor \frac{k_q \tilde{q}_i}{e}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

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 $T_{a,t,\gamma}$ we retain $\tilde{p}_i$ in A, and otherwise we discard $\tilde{p}_i$ from A to reduce the number of options.

Step 3: Suppose that there are x options remaining in array A after the threshold based filtering has been performed. Find the maximum value of c such that c many initial MSBs of all the options are same. 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 $(c+1-t)$-th MSB, and repeat the Guess and Filter routines using the first $(c+a-t)$ MSBs of p.



The Guess and Filter processes are continued until half of p is error-free from the MSB side. After this, one can factorize N using the lattice based idea of Coppersmith [2], as it is done in case of reconstruction as well. The next section presents the formal algorithm for error-correction using all parameters of PK.

3.2 The general error-correction algorithm

In this section, we present the complete algorithm for error-correction of RSA private keys p, q, d, d_p, d_q. Suppose that all bits of PK are known, but each one has a probability γ of being correct. In other words, there is an error probability γ associated with each bit in PK. In such a scenario, Algorithm 3 recovers the top half of the RSA prime p using the idea described above.

```
   Input: $N, e, k, k_p, k_q$, the erroneous approximations $p', q', d'_2, d'_p, d'_q$, , and the
          percentage of error $\gamma$.
   Output: Top half of the MSBs of $p$, corrected.
1  Choose parameters $a, t, b = 0, c_{old} = 1$;
2  Choose threshold $T_{a,t,\gamma}$ depending on $a, t, \gamma$;
3  while $b < \frac{l_N}{2}$ do
4      For first $(a + b)$ MSBs of $p$, choose all possible options $\tilde{p}$ and store in an
       array $A$;
5      $A = \mathsf{Filter}(N, p', q', d'_2, d'_p, d'_q, A, a, t, b, \gamma, T_{a,t,\gamma})$;
6      If the array $A$ remains unaltered, increase $a$ and go to Step 4;
7      if $|A| = 1$ then
8          Set first $(a + b - t)$ MSBs of $\tilde{p} = A[1]$ as those of prime $p$;
9          Set $b = a + b - t$ and $c_{old} = a + b - t$;
       end
10     else
11         Find $c \leq a + b$ such that the first $c$ MSBs of all members of $A$ are same;
12         If $c$ remains unaltered $(c = c_{old})$, decrease $T_{a,t,\gamma}$ and go to Step 4;
13         Set $b = c - t$ and $c_{old} = c - t$;
       end
   end
```

Algorithm 3: CorPK: MSB error-correction algorithm using p, q, d_2, d_p, d_q.

Choice of threshold: One may note that in Step 2 of Algorithm 3, we need to choose a threshold $T_{a,t,\gamma}$ based on a, t and γ. Theoretically, in case with all five RSA parameters p, q, d, d_p, d_q, one should have a threshold $T_{a,t,\gamma} \geq 5(a - t)\gamma$ to accommodate for the correct values to be within the search range. However, as we use the Hamming distance for $5a$ bits for filtering, the threshold should be increased to $T_{a,t,\gamma} \geq 5a\gamma$.

```
Input: $N$, the erroneous approximations $p', q', d'_2, d'_p, d'_q$, array $A$, parameters
       $a, t, b$, and the percentage of error $\gamma$.
Output: Modified array $A$ of bit strings.
1  for $i = 1$ to $|A|$ do
2      $\tilde{p} = A[i]$;
3      Calculate $\tilde{q} = \lfloor \frac{N}{\tilde{p}} \rfloor$, $\tilde{d_2} = \lfloor \frac{k(N - \tilde{p} - \tilde{q})}{e} \rfloor$, $\tilde{d_p} = \lfloor \frac{k_p \tilde{p}}{e} \rfloor$, $\tilde{d_q} = \lfloor \frac{k_q \tilde{q}}{e} \rfloor$;
4      Calculate the Hamming distance of block of MSBs $[b, (a + b)]$ of $p'$ with $\tilde{p}$,
       $q'$ with $\tilde{q}$, $d'_2$ with $\tilde{d_2}$, $d'_p$ with $\tilde{d_p}$, and $d'_q$ with $\tilde{d_q}$ respectively;
5      If the sum of all Hamming distances is more than threshold $T_{a,t,\gamma}$, discard
       corresponding $\tilde{p}$ from $A$;
   end
6  Return $A$;
```
Algorithm 4: Subroutine Filter used in CorPK algorithm.

In practice, we start with a very high value of the threshold $T_{a,t,\gamma} \approx 5a(0.5 + \gamma)$, and gradually decrease it during runtime in each step where the sliding-window approach does not provide a new pivot point c (as in Step 12 of Algorithm 3). Similarly, we choose $T_{a,t,\gamma} \approx 2a(0.5 + \gamma)$ in the p, q case, and $T_{a,t,\gamma} \approx 3a(0.5 + \gamma)$ in the p, q, d case. The initial choices of $T_{a,t,\gamma}$ are verified and finalized through extensive experimentation.

3.3 Experimental results for error-correction

We support our claim with the help of some representative cases of practical reconstructions. The average performance of the algorithm over 100 runs in each of the representative cases is presented in Table 1.

We have implemented using C programming language (with GMP library for processing large integers) on Linux Ubuntu 11.04. The hardware platform is an HP Z800 workstation with 3GHz Intel(R) Xeon(R) CPU. In these experiments, we have taken 1024 bit N, and $e = 2^{16} + 1$ in all cases. The average performance of the algorithm over 100 runs in each case is presented in Table 1.

In case of practical experiments with our idea for error-correction, we have successfully corrected approximately 6% bit error in the p, q case, 11% bit error in the p, q, d case, and 18% bit error in the case with all the PK parameters p, q, d, d_p, d_q. In the last case, 18% error could be corrected using $a = 16$, offset $t = 3$ and an initial threshold $T_{a,t,\gamma} = 54$.

Comparison with previous works: Note that our result is competitive with the results published in [5] where the experimental values were 8%, 14% and 20% in the three respective cases where the algorithms proceeded from the least significant side of the secret parameters. Further, our work is the first in studying the scenario from the most significant side and the algorithms are completely different from that of [5].

Table 1. Experimental data for error-correction algorithm CorPK.

Erroneous approximations of p and q known					
a	t	Threshold $T_{a,t,\gamma}$	Error γ (%)	Success probability	Time T (sec)
12	3	12	3	0.34	4.0
		12	4	0.15	4.2
14	3	14	3	0.39	12.2
		15	4	0.22	12.4
16	3	16	3	0.59	38.5
		17	4	0.41	42.8
		17	5	0.11	43.2
18	3	19	3	0.74	216.3
		19	4	0.42	224.8
		19	5	0.24	231.2
		19	6	0.10	235.6

Erroneous approximations of p, q and d known					
a	t	Threshold $T_{a,t,\gamma}$	Error γ (%)	Success probability	Time T (sec)
12	3	20	7	0.23	7.9
		20	8	0.10	8.4
14	3	23	7	0.48	20.1
		24	8	0.26	21.1
		24	9	0.20	21.9
		25	10	0.10	22.4
16	3	27	7	0.66	62.2
		27	8	0.58	62.6
		28	9	0.31	62.8
		28	10	0.17	59.0
		29	11	0.10	59.9

Erroneous approximations of p, q, d, d_p and d_q known					
a	t	Threshold $T_{a,t,\gamma}$	Error γ (%)	Success probability	Time T (sec)
12	3	38	14	0.28	21.8
		39	15	0.13	21.8
14	3	44	14	0.44	60.7
		45	15	0.32	57.1
		46	16	0.15	58.3
		46	17	0.10	53.5
16	3	51	14	0.69	167.0
		52	15	0.46	165.5
		52	16	0.29	168.1
		53	17	0.19	162.6
		54	18	0.12	163.1

4 Conclusion

In this paper, we study error-correction in RSA private key PK when all the bits are known with some probability of error. We propose a new strategy for correcting the bit-errors in RSA private key, starting from the MSB side of the parameters. Suppose that all the bits of the private key PK are known, but with some probability of error γ for each bit. Our experimental results show that one can correct the errors in the private key PK efficiently when the error rate is

- $\gamma \leq 0.18$ when p, q, d, d_p, d_q are known with error, or
- $\gamma \leq 0.11$ when p, q, d are known with error, or
- $\gamma \leq 0.06$ when p, q are known with error.

Future scope: As a future direction of research, it will be nice to look into the theory behind reconstruction and error-correction. Improved algorithms, both from least and most significant side of the secret parameters, will be of interest to the community.

Acknowledgments. The authors would like to thank the anonymous reviewers for their valuable comments that helped improve the quality of the paper.

References

1. D. Boneh, G. Durfee, and Y. Frankel. An attack on RSA given a small fraction of the private key bits. In *Proceedings of Asiacrypt'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 25–34, 1998.
2. D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.
3. S. W. Graham and I. E. Shparlinski. On RSA moduli with almost half of the bits prescribed. *Discrete Applied Mathematics*, 156(16):3150–3154, 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. *Commun. ACM*, 52(5):91–98, 2009.
5. W. Henecka, A. May, and A. Meurer. Correcting errors in RSA private keys. In *Proceedings of Crypto'10*, volume 6223 of *Lecture Notes in Computer Science*, pages 351–369, 2010.
6. N. Heninger and H. Shacham. Reconstructing RSA private keys from random key bits. In *Proceedings of Crypto'09*, volume 5677 of *Lecture Notes in Computer Science*, pages 1–17, 2009.
7. M. Herrmann and A. May. Solving linear equations modulo divisors: On factoring given any bits. In *Proceedings of Asiacrypt'08*, volume 5350 of *Lecture Notes in Computer Science*, pages 406–424, 2008.
8. N. Howgrave-Graham. Finding small roots of univariate modular equations revisited. In *Proceedings of IMA International Conference on Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 131–142, 1997.
9. E. Jochemsz and A. May. A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants. In *Proceedings of Asiacrypt'06*, volume 4284 of *Lecture Notes in Computer Science*, pages 267–282, 2006.

10. A. K. Lenstra. Generating RSA moduli with a predetermined portion. In *Proceedings of Asiacrypt'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 1–10, 1998.
11. A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.
12. S. Maitra, S. Sarkar, and S. Sen Gupta. Factoring RSA modulus using prime reconstruction from random known bits. In *Proceedings of Africacrypt'10*, volume 6055 of *Lecture Notes in Computer Science*, pages 82–99, 2010.
13. K. G. Paterson, A. Polychroniadou, D. L. Sibborn. A Coding-Theoretic Approach to Recovering Noisy RSA Keys. In *Proceedings of Asiacrypt'12*, volume 7658 of *Lecture Notes in Computer Science*, pages 386–403, 2012.
14. Public-Key Cryptography Standards (PKCS) #1 v2.1: RSA Cryptography Standard. RSA Security Inc., 2002. Available at `http://www.rsa.com/rsalabs/node.asp?id=2125`.
15. J. J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronic Letters*, 18(21):905–907, 1982.
16. R. L. Rivest and A. Shamir. Efficient factoring based on partial information. In *Proceedings of Eurocrypt'85*, volume 219 of *Lecture Notes in Computer Science*, pages 31–34, 1986.
17. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.
18. S. Sarkar, S. Sen Gupta, and S. Maitra. Reconstruction and error correction of RSA secret parameters from the MSB side. In *Workshop on Coding and Cryptography*, 2011.
19. I. E. Shparlinski. On RSA moduli with prescribed bit patterns. *Designs, Codes and Cryptography*, 39:113–122, 2006.