Better Algorithms for MSB-side RSA Reconstruction

Pratibha Jagnere, Srijan Sanket, Anuj Chauhan, and Ragesh Jaiswal*

Department of Computer Science and Engineering,
Indian Institute of Technology Delhi.

Abstract. In this work, we extend research in the area of reconstruction of private RSA keys using few random bits made available through partial key exposure attacks such as the *Cold Boot Attack*. Our work discusses an approach to reconstruct private components of RSA from the *most significant bit (MSB) side* given only 26% of the secret bits. Our approach shows significant improvement over previous works on reconstruction from the MSB side. Our results closely match those of the Heninger and Shacham that gives reconstruction from the LSB side.

1 Introduction

In this work, we discuss reconstruction of secret keys of the RSA encryption scheme. The basic RSA encryption scheme can be described as follows: $Gen(1^n)$ first chooses two n-bit prime numbers p and q and computes $N = p \cdot q$. Let e, d be two integers such that $e \cdot d \equiv 1\ mod(\psi(N))$, where $\psi(N) = (p-1) \cdot (q-1)$. Finally, it outputs the key pair (sk, pk), where $sk = (d, p, q)$ and $pk = (e, N)$. The encryption algorithm is given by $Enc_{pk}(m) = m^e\ (mod\ N)$ and decryption algorithm is given by $Dec_{sk}(c) = c^d\ (mod\ N)$. The decryption speed of RSA become slower when size of d and N increase. To overcome this problem, a faster variant called CRT-RSA was proposed by Quisquater and Couvreur [QC82]. CRT-RSA is standard way of implementing RSA and also part of PKCS#1. In this variant, there are two additional private exponents dp, dq defined as $dp = (d\ mod\ (p-1))$ and $dq = (d\ mod\ (q-1))$. So (p, q, d, dp, dq) becomes private key. We decrypt the message m given ciphertext c in following way,

$$qp = q^{-1}\ (mod\ p) \quad m_1 = c^{dp}\ (mod\ p) \quad m_2 = c^{dq}\ (mod\ q)$$
$$h = qp \cdot (m_1 - m_2)\ (mod\ p) \quad m = m_2 + h \cdot q$$

In this work, we extend research in the area of reconstruction of private RSA keys using few random bits made available through partial key exposure attack such as the Cold Boot Attack [HSH+09]. There has been significant amount of work in the area of partial key exposure based attacks on cryptographic protocols. The main question that is addressed in this work is the following: *if certain bits of the secret keys of protocols like RSA are known due to side channel attacks, then is it possible to reconstruct the entire secret key?* This questions has been addressed in various abstract settings under various assumptions on the known bits.

The work of Halderman *et al.* [HSH+09] give an interesting practical scenario where certain *random* bits of the secret keys can be made available. The side channel attack that makes this possible is based on data remanence property of the of the typical DRAM memory available on most computers today. The property is that when the power supply to a computer is disconnected, the contents of the memory are not lost immediately but decay with time.[1] The rate of decay is dependent on time and temperature. In the Cold Boot Attack, the attacker gains physical access to the victim's computer and it is set to reboot without clean shutdown (using reset button if available) so that before reboot, the operating system dumps the memory state in a system file which may include private components of the RSA key. After that, the physical memory of the computer is retrieved and attached to the attacker's computer where it can access the memory and file

* The research was done by the authors during their stay at IIT Delhi.
[1] Note that certain bits decay from 0 to 1 while others decay from 1 to 0.

dump, and retrieve the random bits of the private key. Since data starts to disappear from physical memory or RAM, an environment of low room temperature is used to slow down decaying of data.

In their attack, [HSH+09] provide methods to use decayed state data of DES, AES, RSA private keys and use it efficiently to recover the secret key. However, the amount of known bits for their reconstruction algorithm to work is as high as 90% for RSA key reconstruction. Heninger and Shacham [HS09] give a reconstruction algorithm for RSA that works even when as small as 27% of the bits are known. The reconstruction algorithm performs the reconstruction from the Least Significant Bit (LSB) side. This means that that the algorithm guesses the LSB bit first and use this guess to further guess the second bit from LSB and so on. Maitra *et al.* [MSSG10] give a reconstruction algorithm from the MSB side but under the additional assumption that block of bits are known rather than bits. Under this assumption, their reconstruction algorithm works when 41% of the bits are known. Sarkar *et al.* [SGM11, SGM12] extend the previous work and give a reconstruction algorithm that reconstructs from the MSB side without the assumption that blocks of bits are known. In this work, we extend work of Sarkar *et al.* [SGM11, SGM12] and give an algorithm that reconstructs from the MSB side but works even when the fraction of known bits is 26%. Note that this closely matched the results of Heninger and Shacham [HS09]. We also give a theoretical analysis of our algorithm and show that under reasonable assumptions, the running time of our MSB side reconstruction algorithm is not more than some constant times that of the LSB side reconstruction algorithm of Heninger and Shacham [HS09].

1.1 Related Work

An efficient algorithm for solving the factoring problem (given $N = p \cdot q$, find p and q) would imply breaking the RSA scheme. Fortunately, all known algorithms for factoring are highly inefficient. However, there have been successful attempts of solving weaker versions of the factoring problem where p and q are partially known. These have been argued to be interesting in cases where partial knowledge of p and q is made possible through certain side channel attacks. Rivest and Shamir [RS86] showed that an RSA modulus N could be factored given 2/3 of the bits of the most or least significant bits of one of the primes. Coppersmith reduced the fraction of needed bits to 1/2 in [Cop96] when the most significant bits are known. The main idea was to find a unique solution of the bivariate polynomial $(x + \tilde{p}) \cdot (y + \tilde{q}) = N$, where $\tilde{p}$ and $\tilde{q}$ correspond to the partial information regarding p and q. The known least significant bits case was later observed by Boneh *et al.* [BDF98]. [HM08] consider the more general problem where instead of the block of first/second half bits known, any number of blocks of bits of one of the primes are known for a smaller block size. They show that knowing $\log_e(2) \approx 0.7$ of the bits is sufficient to factor the modulus. However, the complexity of their method is exponential in the number of blocks and so the number of blocks must be $O(\log \log N)$ for polynomial running time.

The above were factoring based attacks on RSA. Here the attacker is trying to factor N to obtain p and q. Note that this reveals the private exponent d. *Partial key exposure attacks* are attacks on RSA where the private exponent d is partially known. These attacks have attracted many researchers into studying the physical implementation of RSA. The first known partial key exposure attack for RSA with known bits from the MSB/LSB side of the private exponent was given by Boneh *et al.* [BDF98]. Their algorithm reconstructs the private exponent d given a quarter of its bits from the least significant bit side for any public exponent $\leq N^{1/2}$. They also give attacks when bits are known from the MSB side. Blömer and May [BM03] gave partial key exposure attacks for larger values of the public exponent but still not to the full size of the modulus. Ernst *et al.* [EJMW05] gave attacks that work for all values of the public exponent. Aono [Aon09] improved it for small values of the private exponent d.

All of the above attacks require that a contiguous block of the most or least significant bits should be known. The cold boot attack of Halderman *et al.* [HSH+09] introduced a new scenario that was not discussed by any of this previous works. The attack makes non-contiguous bits of the secret keys known instead of contiguous blocks of bits. The simple way to model the problem is to assume that some δ fraction of the bits of the

secret keys are known and these known bits are distributed randomly within the secret string. Halderman *et al.* [HSH+09] gave simple reconstruction algorithms for RSA along with other primitives such as DES, AES etc. However, the value of δ for which their reconstructions works was high. Heninger and Shacham [HS09] gave an improved algorithm that works for even smaller values of δ. They also gave a theoretical analysis showing a bound on the number of keys examined by their algorithm that depends on the value of δ. Both the above reconstruction algorithms work from the LSB side. That is, the algorithm first tries to guess the bits from the LSB side and then guesses the lower precision bits. Maitra *et al.* [MSSG10] gave a reconstruction algorithm that works from the MSB side. However, they assume that blocks of bits are known instead of bits. A reconstruction algortihm given by Sarkar *et al.* [SGM11, SGM12] works from the MSB side and on known bits. However, their algorithm works for values of δ that is larger than those in Heninger and Shacham. Paterson *et al.* [PPS12] gave a coding theoretic approach to recover RSA keys that are noisy. Sarkar and Maitra [SM12] discuss reconstruction algorithms for recovering noisy RSA keys when the decryption exponent has low Hamming weight.

In our work, we give a reconstruction algorithm that works from the MSB side and gives results for same values of δ as in Heninger and Shacham [HS09]. We give a theoretical analysis on the lines of [HS09]. We discuss our contributions in the next subsection.

1.2 Our Contributions

The general goal of our project is to reconstruct bit strings given a *partial version* of the string. First, we need to define clearly what we mean by a *partial version* of a string. Given a parameter $0 < \delta \leq 1$ and an n-bit string s, let s_δ denote a random string in $\{0, 1, *\}^n$ defined in the following manner: for all i, $s_\delta[i]$ is equal to $s[i]$ with probability δ and $*$ with remaining probability. All the indices are independently sampled. Let $\mathcal{D}(\delta, s)$ denote this distribution over $\{0, 1, *\}^n$. Sampling a string from this distribution is denoted by $\tilde{s} \leftarrow \mathcal{D}(\delta, s)$. The above model captures the random decay of bits in the cold boot attack. We consider the following three versions of the RSA reconstruction problem. For all three versions we consider the standard RSA parameters (N, p, q, e, d). For CRT-RSA we also have $dp = (d \ mod \ (p-1))$ and $dq = (d \ mod \ (q-1))$.

Factoring	**RSA**	**CRT-RSA**
Inputs:	*Inputs*:	*Inputs*:
$\tilde{p} \leftarrow \mathcal{D}(\delta, p)$	$\tilde{p} \leftarrow \mathcal{D}(\delta, p)$	$\tilde{p} \leftarrow \mathcal{D}(\delta, p)$
$\tilde{q} \leftarrow \mathcal{D}(\delta, q)$	$\tilde{q} \leftarrow \mathcal{D}(\delta, q)$	$\tilde{q} \leftarrow \mathcal{D}(\delta, q)$
	$\tilde{d} \leftarrow \mathcal{D}(\delta, d)$	$\tilde{d} \leftarrow \mathcal{D}(\delta, d)$
		$\tilde{dp} \leftarrow \mathcal{D}(\delta, dp)$
		$\tilde{dq} \leftarrow \mathcal{D}(\delta, dq)$
Output:	*Output*:	*Output*:
p or q	p or q or d	p or q or d

Our two main contributions in this work are as follows:

1. We get an improvement in the value of δ (i.e., the fraction of known bits) for which reconstruction can be done in reasonable amount of time (a few minutes) from the MSB side. Table 1 gives a snapshot of our work in comparison with previous works.[2]
2. We give a theoretical analysis on the lines of the analysis in Heninger and Shacham [HS09]. We show that under reasonable assumptions, for the same values of δ and n, the number of keys examined by our MSB side reconstruction algorithm will be at most some fixed constant times the number of keys examined by the LSB side reconstruction algorithm of Heninger and Shacham [HS09].

[2] Note that the improvements over Sarkar *et al.* [SGM11, SGM12] comes from the fact that we are able to prove a theoretical bound of a parameter (t) that is fixed experimentally in [SGM11, SGM12].

	Factoring	RSA	CRT-RSA
Halderman *et al.* [HSH+09]	-	-	0.88
Heninger and Shacham [HS09] (LSB reconstruction)	0.53	0.38	0.23
Sarkar *et al.* [SGM11, SGM12] (MSB reconstruction)	0.61	0.46	0.38
This work (MSB reconstruction)	0.53	0.48	0.26

Table 1. This table shows that value of δ for which the reconstruction algorithm works on 512-bit primes in reasonable amount of time (a few minutes).

Organisation This paper is divided into four sections. In Section 2, we present the main ideas and give the design and analysis of our reconstruction algorithms. Section 3 gives the results of the empirical analysis that we conducted.

2 Reconstruction Algorithms

In this section, we describe our reconstruction algorithms. We consider the following three cases:

1. Factoring: When partial bits of p and q are given.
2. RSA: When partial bits of p, q, and d are given.
3. CRT-RSA: This is with respect to the CRT-RSA implementation described in the previous section. Here, partial bits of p, q, d, dp, dq are known.

We describe our reconstruction algorithms for each of these cases in the next three subsections.

2.1 Reconstruction using partially known p, q

In this subsection, we present our reconstruction algorithm that uses branch-and-bound to reconstruct from the MSB side. To be able to understand the algorithm better, we first present some simple properties of binary numbers. We need to define the following quantities: The i^{th} bit of an n bit binary number r is denoted by $r[i]$. The first k bits of r from the MSB side, i.e., $r[n-1], r[n-2], ..., r[n-k]$ is denoted by r_k. That is, we have $r_k = \sum_{i=0}^{k-1} 2^i \cdot r[n-k+i]$.

Also, for $i < j$, let $r_{j,i}$ denote the number corresponding to the bits $(r[j], r[j-1], ..., r[i])$, i.e., $r_{j,i} = \sum_{k=0}^{j-i-1} 2^k \cdot r[i+k]$. In our discussion, whenever we multiply two number of size say α and β bits, then the resulting multiple is considered to be of size $(\alpha + \beta)$ (Note that there might be leading zeros in the numbers and their multiple). The next two lemmas would be the defining results for our approach.

Lemma 1. *Let p, q be n bit primes and let $N = p \cdot q$ and let $k > 0$. Let $T = p_k \cdot q_k$. Then*

$$N_k - T_k \leq 2.$$

Proof. Let $a = p_{n-k-1,0}$ and $b = q_{n-k-1,0}$. We can write p and q as follows:

$$p = 2^{n-k} \cdot p_k + a \quad \text{and} \quad q = 2^{n-k} \cdot q_k + b \tag{1}$$

Using this, N may be written as:

$$N = p \cdot q = 2^{2n-2k} \cdot T + 2^{n-k} \cdot (p_k \cdot b) + a \cdot q \tag{2}$$

Performing an integer division on both sides of the above equation by 2^{2n-k}, we get the following:

$$N_k = T_k + \lfloor \left(\frac{2^{2n-2k} \cdot T_{k-1,0} + 2^{n-k} \cdot (p_k \cdot b) + a \cdot q}{2^{2n-k}} \right) \rfloor \tag{3}$$

Now, we know the following:

$$\begin{aligned} 2^{2n-2k} \cdot T_{k-1,0} &\leq 2^{2n-2k} \cdot (2^k - 1) < 2^{2n-k}, \\ 2^{n-k} \cdot (p_k \cdot b) &\leq 2^{n-k} \cdot (2^k - 1) \cdot (2^{n-k} - 1) < 2^{2n-k}, \\ a \cdot q &\leq (2^{n-k} - 1) \cdot (2^n - 1) < 2^{2n-k} \end{aligned}$$

Using the above three inequalities and (3), we get statement of the Lemma. □

The next lemma shows that we can get a close estimate of q_k given the correct value of p_k.

Lemma 2. *Let p, q be n bit prime numbers. Let $k > 0$. Let R be a $2k$ bit number defined in the following manner:*

$$R = (N_k - 1) \cdot 2^k + 2^{k-1}.$$

Then $|q_k - \lfloor R/p_k \rfloor| \leq 3$.

Proof. Let $T = p_k \cdot q_k$. We may write T as:

$$\begin{aligned} T &= 2^k \cdot T_k + T_{k-1,0} \\ &= 2^k \cdot (N_k - (N_k - T_k)) + T_{k-1,0} \end{aligned}$$

We use this to get the following:

$$\begin{aligned} |T - R| &= |2^k \cdot (1 - (N_k - T_k)) + T_{k-1,0} - 2^{k-1}| \\ &\leq 2^k \cdot |(1 - (N_k - T_k))| + |T_{k-1,0} - 2^{k-1}| \\ &\leq 2^k + |T_{k-1,0} - 2^{k-1}| \quad \text{(using Lemma 1)} \\ &\leq 2^k + 2^{k-1} \end{aligned}$$

Dividing both sides by p_k we get the following:

$$\begin{aligned} |T/p_k - R/p_k| &\leq \frac{2^k + 2^{k-1}}{p_k} \\ \Rightarrow |q_k - R/p_k| &\leq \frac{2^k + 2^{k-1}}{p_k} \quad (\text{since } T = p_k \cdot q_k) \\ &\leq 3 \quad (\text{since } p_k \geq 2^{k-1}) \\ \Rightarrow |q_k - \lfloor R/p_k \rfloor| &\leq 3. \end{aligned}$$

The last inequality is due to the fact that the first bit of p and q are 1. □

The above lemma shows that for the correct value of p_k, we can get the approximate value of q_k by dividing a number R (that depends only on N) by p_k. The value obtained may be off by an additive factor of 3. This means that the possible values of q_k are $\lfloor R/p_k \rfloor - 3, \lfloor R/p_k \rfloor - 2, \lfloor R/p_k \rfloor - 1, \lfloor R/p_k \rfloor, \lfloor R/p_k \rfloor + 1, \lfloor R/p_k \rfloor + 2, \lfloor R/p_k \rfloor + 3$. Our algorithm is a backtracking algorithm that reconstructs p and q from the MSB side. Suppose p'_i is the current value of the first i bits of p that the algorithm is trying. The algorithm tries out both 0/1 values of $p'[i+1]$ (only one if $p[i+1]$ is known). For each value, it computes all 7 candidate values of q'_{i+1} using the above lemma. If some values of q'_{i+1} is not consistent with the value of q'_i computed previously (or with $q[i+1]$ if it is known), then these values of q'_{i+1} are discarded. This algorithm can be written down as a recursive procedure. Algorithm 2.1 gives this recursive procedure. Figure 3 given in the Appendix, illustrates a run of the algorithm for reconstruction of two 6 bit primes.

Inputs:
- N: *//The number to be factored.*
- p, q: *//Arrays that contains partially known bits of p, q.*
- *//If the i^{th} bit of say p is not known, then $p[i] = \perp$.*

Backtrack-factor(r, S, i)
01. If $(p[i] \neq \perp$ and $p[i] \neq r[i])$ then return.
02. $R \leftarrow 2^k \cdot (N_k - 1) + 2^{k-1}$.
03. $T \leftarrow \lfloor R/r \rfloor$
04. $S' \leftarrow \{T-3, T-2, T-1, T, T+1, T+2, T+3\}$
05. For all $s \in S'$
    If s is inconsistent with S, then $S' \leftarrow S' \setminus \{s\}$.
06. If $(q[i] \neq \perp)$ then for all $s \in S'$
    If $(s[i] \neq q[i])$, then $S' \leftarrow S' \setminus \{s\}$.
07. For all $s \in S'$
    If $(s > r)$, then $S' \leftarrow S' \setminus \{s\}$.
08. For all $s \in S'$
    If $(s < 2^{i-1})$, then $S' \leftarrow S' \setminus \{s\}$.
09. If $S' = \{\}$, then return
10. If $(i = n$ and $\exists s \in S', r \cdot s = N)$,
    then output(**factors found!**)
11. else
    (a) If $(i = n)$ then return
    (b) **Backtrack-factor**$(2 \cdot r, S', i+1)$
    (c) **Backtrack-factor**$(2 \cdot r + 1, S', i+1)$

Algorithm 2.1: Recursive procedure for reconstructing p and q from the MSB side.

Factoring analysis Note that our backtracking algorithm is different from that of Heninger and Shacham where the reconstruction is done from the LSB side. In the remainder of this paper, we will abbreviate the work of Heninger and Shacham as HS. The crucial difference is in the fact that in HS algorithm, knowing the $(i+1)^{th}$ slice of bits (from the LSB side) of p, uniquely fixes the $(i+1)^{th}$ slice of bits of q. On the other hand, in our algorithm, knowing the $(i+1)^{th}$ slice of bits (from the MSB side) of p does not fix the $(i+1)^{th}$ slice of bits of q but there are 7 different possibilities to choose from. This changes the branching structure of the algorithm. [3]

Doing an analysis as done in Heninger and Shacham using the new branching structure does not give encouraging results. So, we will have to analyze our algorithm more closely. Note that the difference in the branching structure is when a bit of q is known: In the HS algorithm branching, this either fixes the current slice of p (in case the corresponding bit of p is not known) or terminates the current branch of execution if there is inconsistency with the known bit of p. Our algorithm branches in both cases and hence have extra branches. We argue that these extra branches are not very deep and they do not have too many execution nodes. This means that in comparison with the HS branching, our algorithm examines more cases, but only by a small multiplicative factor.

[3] In the terminology of Heninger and Shacham's work, the branching structure is captured by the following:

$$EZ_g = (1-\delta)^2 \quad \text{and} \quad EW_b = \frac{(2-\delta)^2}{2} \quad \text{(Heninger-Shacham)}$$
$$EZ_g = (1-\delta) \quad \text{and} \quad EW_b = (2-\delta) \quad \text{(Our Algorithm)}$$

Note that δ denotes the fraction of the bits that are known.

Next, we analyze the depth and size of these extra branches. To be able to do that, we have to use the following conjecture which we empirically observe to be true.

Conjecture 1. Let p, q be randomly chosen n bit numbers (the MSB in p, q is 1) and let $N = p \cdot q$. Let p'_i be any i-bit number such that $p'_i \neq p_i$. Let $T = (N_i - 1) \cdot 2^i + 2^{i-1}$ and $R = \lfloor T/p'_i \rfloor$. Then the following is true:

$$\forall j \leq (i-10),\ \mathbf{Pr}[\forall k, -3 \leq k \leq 3, (R+k)[j] = 0] \approx \mathbf{Pr}[\forall k, -3 \leq k \leq 3, (R+k)[j] = 1] \approx 1/2.$$

Note that probability is over random n-bit primes p, q such that $p'_i \neq p_i$.

Let us now analyze the extra branch at the k^{th} bit when $q[k]$ is known. Now, in the extra branch the slice p'_k is incorrect (i.e., $p'_k \neq p_k$). We will compute the probability that node labeled p'_{k+t} exists where the first k bits of p'_{k+t} are the same as p'_k. Let the set of such p'_{k+t} at depth t be denoted by S. We know that $|S| = 2^t$. Note that all elements of S do not survive until depth t since some of the bits of p at indices between k and $k+t$ are known. Let $S' \subseteq S$ such that $\forall e \in S'$ e is consistent with the known bits in p. We will compute the expectation of S'. The probability that m bits are known is given by $\binom{t}{m}\delta^m(1-\delta)^{t-m}$. Given that m bits are known the size of S' is given by 2^{t-m}. So we get that:

$$\begin{aligned} E[S'] &= \sum_{m=0}^{t} \binom{t}{m} \delta^m (1-\delta)^{t-m} \cdot 2^{t-m} \\ &= (2-\delta)^t \end{aligned}$$

Consider any element of the $p'_{k+t} \in S'$. Now, p'_{k+t} might not survive because the corresponding 7 choices for q'_{k+t} are not consistent with the known bits in q. Using the conjecture above we get that:

$$\begin{aligned} \mathbf{Pr}[p'_{k+t} \text{ survives}] &\leq \sum_{m=0}^{t-10} \binom{t-10}{m} \delta^m (1-\delta)^{t-10-m} \cdot 2^{-m} \\ &= (1-\delta/2)^{t-10} \end{aligned}$$

Let S'' denote the surviving nodes at depth t. Using the above inequalities, we get that

$$\begin{aligned} E[S''] &= (2-\delta)^t \cdot (1-\delta/2)^{t-10} \\ &= \frac{1}{(1-\delta/2)^{10}} \cdot \left(\frac{(2-\delta)^2}{2}\right)^t \end{aligned}$$

Note that when $\delta = 2 - \sqrt{2} = 0.5857$, then $E[S''] \leq 2^{10}$. This means that the expected number of cases examined by our branching algorithm is at most $(2^{10} \cdot n)$ times the expected number of cases examined by the HS branching algorithm.

2.2 Reconstruction using partially known p, q, d

Most commonly deployed RSA systems use small public exponent. It has been observed that the public exponent e in most applications is of length 32 bits and in fact using $e = 65537 = 2^{16} + 1$ is very common. We use this, fact to design a reconstruction algorithm when p, q, and d are known partially. First, we see how we can use the public parameter e and N to compute some useful quantities. Recall that d is the inverse of e modulo $\psi(N)(= (p-1) \cdot (q-1))$. This gives us the following:

$$e \cdot d = k \cdot \psi(N) + 1 \tag{4}$$

where $k > 0$ is some positive integer. Note that $k < e$ since $d < \psi(N)$. This means if the RSA implementation uses a small public exponent e, then we have a to try a small number of value of k while running the

reconstruction algorithm. So, for the rest of the discussion, we will assume that the correct value of k is known (in actual implementation, we will try all possible value of k). Let m denote the length of d. As, in the previous section we denote the first i bits from the MSB side of d as d_i. The following quantity (in terms of k, e, N, p_i, q_i) is a close approximation to d_i.

$$d'_i = \lfloor \frac{k \cdot (N+1)+1}{e \cdot 2^{m-i}} \rfloor - \lfloor \frac{k \cdot (p_i + q_i)}{e \cdot 2^{m-n}} \rfloor \tag{5}$$

The next lemma shows that d'_i is a close approximation to d_i.

Lemma 3. *$\forall i$, If $m \geq n+1$, then $|d_i - d'_i| \leq 3$.*

Proof. Let ρ denote the last (from MSB side) $(n-i)$ bits of p, σ denote the last $(n-i)$ bits of q, ν denote the last $(m-i)$ bits of d. We have the following:

$$\begin{aligned}
& e \cdot d = k \cdot (N - p - q + 1) + 1 \\
\Rightarrow\ & e \cdot (d_i \cdot 2^{m-i} + \nu) = k \cdot (N - p_i \cdot 2^{n-i} - \rho - q_i \cdot 2^{n-i} - \sigma + 1) + 1 \\
\Rightarrow\ & d_i = \frac{k \cdot (N+1)+1}{e \cdot 2^{m-i}} - \frac{k \cdot (p_i + q_i)}{e \cdot 2^{m-n}} - \frac{k \cdot (\rho + \sigma) + e \cdot \nu}{e \cdot 2^{m-i}}
\end{aligned}$$

Using equation (5) we get the following:

$$\begin{aligned}
|d_i - d'_i| &\leq 1 + |\frac{k \cdot (\rho + \sigma) + e \cdot \nu}{e \cdot 2^{m-i}}| \\
&\leq 1 + |\frac{\rho + \sigma + \nu}{2^{m-i}}| \quad (\text{since } k < e) \\
&\leq 1 + \frac{2^{n-i} + 2^{n-i} + 2^{m-i}}{2^{m-i}} \\
&\leq 2 + 2^{n-m+1} \\
&\leq 3 \quad (\text{since } m \geq n+1)
\end{aligned}$$

This concludes the proof of the lemma. □

Note that the last inequality in the above proof uses the fact that $m \geq n+1$. This is true for the case we are considering. i.e., when e is small (say $e \leq 65537$) and follows from equation (4). So, when partial information of p, q, and d are known, then we can modify the reconstruction algorithm of the previous section in the following manner:

> For each value of $k \leq e$, do the following: During the execution of the previous algorithm, for each value of p_i and q_i that the algorithm is trying, compute the value of d'_i using equation (5) and then try all possibilities in the set $\{d'_i - 3, d'_i - 2, d'_i - 1, d'_i, d'_i + 1, d'_i + 2, d'_i + 3\}$.

We can get further improvement in our reconstruction by making the following observation:

> When e is small, then most of the MSB bits from the first half of d depends on just the public parameters such as N, e and k, and m that we can estimate.

This can be seen as follows. From the previous lemma, we know that d'_i is a close approximation to d_i when e is small (and hence $m \geq n+1$). Now lets take a closer look at d'_i below.

$$d'_i = \left\lfloor \frac{k \cdot (N+1)+1}{e \cdot 2^{m-i}} \right\rfloor - \left\lfloor \frac{k \cdot (p_i + q_i)}{e \cdot 2^{m-n}} \right\rfloor = L_i - R_i$$

where

$$L_i = \left\lfloor \frac{k \cdot (N+1)+1}{e \cdot 2^{m-i}} \right\rfloor \qquad \text{and} \qquad R_i = \left\lfloor \frac{k \cdot (p_i + q_i)}{e \cdot 2^{m-n}} \right\rfloor$$

Note that $k \leq e$. Also, if is e is as small as $2^{16}+1$, then $m \geq 2n-r$ for some small constant r. So, for $i \leq n-r-2$ we have $R_i < 1$. This means that d_i depends mainly on L_i which is a function of e, N, m, and k. This in turn means that we may as well figure out the MBS bits of d and use the reconstruction algorithm for the factoring problem discussed in the previous section.

2.3 Reconstruction using partially known p, q, d, dp, dq

As we saw in the introduction, CRT-RSA private keys may be described as an 8-tuple $(N, e, d, p, q, dp, dq, qp^{-1})$. Recall the definition of some of these terms.

- $dp = d \mod (p-1)$ and $dq = d \mod (q-1)$.
- $qp = q \mod p$

We have the following new equalities that relate some of the above quantities:

$$e \cdot dp = k_p \cdot (p-1) + 1 \tag{6}$$

$$e \cdot dq = k_p \cdot (q-1) + 1 \tag{7}$$

Given the correct value of k, the value of k_p and k_q can be computed easily when e is small. See [HS09] for a discussion on this. For the remaining discussion, we will assume that the values of k, k_p, and k_q are known [4]. We will assume that dp and dq are represented using n bits (use padding if necessary). The following give close approximation to the first i bits (from MSB side) of dp and dq, given that the first i bits of p and q are known respectively.

$$dp'_i = \lfloor \frac{k_p \cdot p_i}{e} \rfloor \tag{8}$$

$$dq'_i = \lfloor \frac{k_q \cdot q_i}{e} \rfloor \tag{9}$$

The following lemma show that the above quantities are close approximations to the correct quantities dp_i and dq_i.

Lemma 4. *$\forall i, |dp_i - dp'_i| \leq 2$ and $|dq_i - dq'_i| \leq 2$.*

Proof. Let ρ denote the last $(n-i)$ bits of dp and let σ denote the last $(n-i)$ bits of p_i. Then we have:

$$\begin{aligned} & e \cdot dp = k_p \cdot (p-1) + 1 \\ \Rightarrow\ & e \cdot (dp_i \cdot 2^{n-i} + \rho) = k_p \cdot (p_i \cdot 2^{n-i} + \sigma - 1) + 1 \\ \Rightarrow\ & dp_i = \frac{k_p \cdot p_i}{e} + \frac{k_p \cdot \sigma - e \cdot \rho}{e \cdot 2^{n-i}} - \frac{k_p - 1}{2^{n-i}} \end{aligned}$$

Using equation (8), we get the following:

$$\begin{aligned} |dp_i - dp'_i| &\leq 1 + |\frac{k_p \cdot (\sigma - 1) - e \cdot \rho + 1}{e \cdot 2^{n-i}}| \\ &\leq 1 + |\frac{\sigma - 1 - \rho + 1/e}{2^{n-i}}| \quad (\text{since } k_p < e) \\ &\leq 2 \end{aligned}$$

Note that we have used $k_p < e$. This follows from equation (6) by observing that $dp < (p-1)$. The second part of the Lemma, i.e., $|dq_i - dq'_i| \leq 2$ follows by symmetry. □

[4] In practice, what we will have is a short list of candidates for k, k_p, and k_q and the algorithm will have to try all of them to be able to find p and q

We summarize this section outlining the modifications to be made in the reconstruction algorithm when partial information for p, q, d, dp, dq is known.

> For each value of $k < e$, compute the prospective values of k_p and k_q. For each combination of k, k_p, k_q do the following: For each value of p_i and q_i that the algorithm is trying, compute the values d'_i using equation (5), dp'_i using equation (8), and dq'_i using equation (9). Then try 7 possibilities $\{d'_i - 3, d'_i - 2, d'_i - 1, d'_i, d'_i + 1, d'_i + 2, d'_i + 3\}$ for d_i, 5 possibilities $\{dp'_i - 2, dp'_i - 1, dp'_i, dp'_i + 1, dp'_i + 2\}$ for dp_i and 5 possibilities $\{dq'_i - 2, dq'_i - 1, dq'_i, dq'_i + 1, dq'_i + 2\}$ for dq_i.

3 Empirical Analysis

We have implemented of our key reconstruction algorithms in Python 2.7.2, using Pycrypto 2.6 library on Ubuntu 12.04 (precise) 64-bit, kernel - 3.2.0-23. Our experiments were run on Intel Core $i7$-2600S CPU @ 2.80 GHzx8 with 8GB Non-ECC DDR3 1600MHz SDRAM memory. The code may be found at the following link: http://www.cse.iitd.ac.in/ rjaiswal/Research/Cold-boot/RSA-reconstruction.zip.
In these experiments, we use key size varying from 512 bits to 2048 bits and $e = 2^{16} + 1$ in all the cases. If the running time of an execution exceeds 5 minutes, the program terminates without giving results otherwise the secret keys are outputted. The number of keys examined before finding the result is also recorded. We create boxplot for both conditions i.e. when we run the algorithm to reconstruct from MSB side only and when we run it with Heninger and Shacham's algorithm in [HS09]. We examine the boxplot to study the behavior of algorithm for different values of δ.
We will start with giving comparison of the number of keys examined in the form of boxplot with respect to the boxplot given by Heninger-Shacham [HS09]. The range of δ is same for both the boxplots and N is $2048 - bits$. Figure 1 shows the values given by Heninger and Shacham whereas Figure 2 gives the values given by our MSB side only recursive algorithm. Note that empirical results validates the theoretical bounds observed in the previous section. The observation was that the number of keys examined by our MSB-side algorithms will not be much more than some constant factor times the number of keys examined by the LSB-side algorithm of Heninger and Shacham [HS09].

References

[Aon09] Yoshinori Aono. A new lattice construction for partial key exposure attack for RSA. In *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography: PKC '09*, Irvine, pages 34–53, Berlin, Heidelberg, 2009. Springer-Verlag.

[BDF98] Dan Boneh, Glenn Durfee, and Yair Frankel. An attack on RSA given a small fraction of the private key bits. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology ASIACRYPT'98*, volume 1514 of *Lecture Notes in Computer Science*, pages 25–34. Springer Berlin Heidelberg, 1998.

[BM03] Johannes Blömer and Alexander May. New partial key exposure attacks on RSA. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 27–43. Springer Berlin Heidelberg, 2003.

[Cop96] Don Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. In *Proceedings of the 15th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'96, pages 178–189, Berlin, Heidelberg, 1996. Springer-Verlag.

[EJMW05] Matthias Ernst, Ellen Jochemsz, Alexander May, and Benne Weger. Partial key exposure attacks on RSA up to full size exponents. In Ronald Cramer, editor, *Advances in Cryptology EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 371–386. Springer Berlin Heidelberg, 2005.

[HM08] Mathias Herrmann and Alexander May. Solving linear equations modulo divisors: On factoring given any bits. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 406–424. Springer Berlin Heidelberg, 2008.

[HS09] Nadia Heninger and Hovav Shacham. Reconstructing RSA private keys from random key bits. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg, 2009.

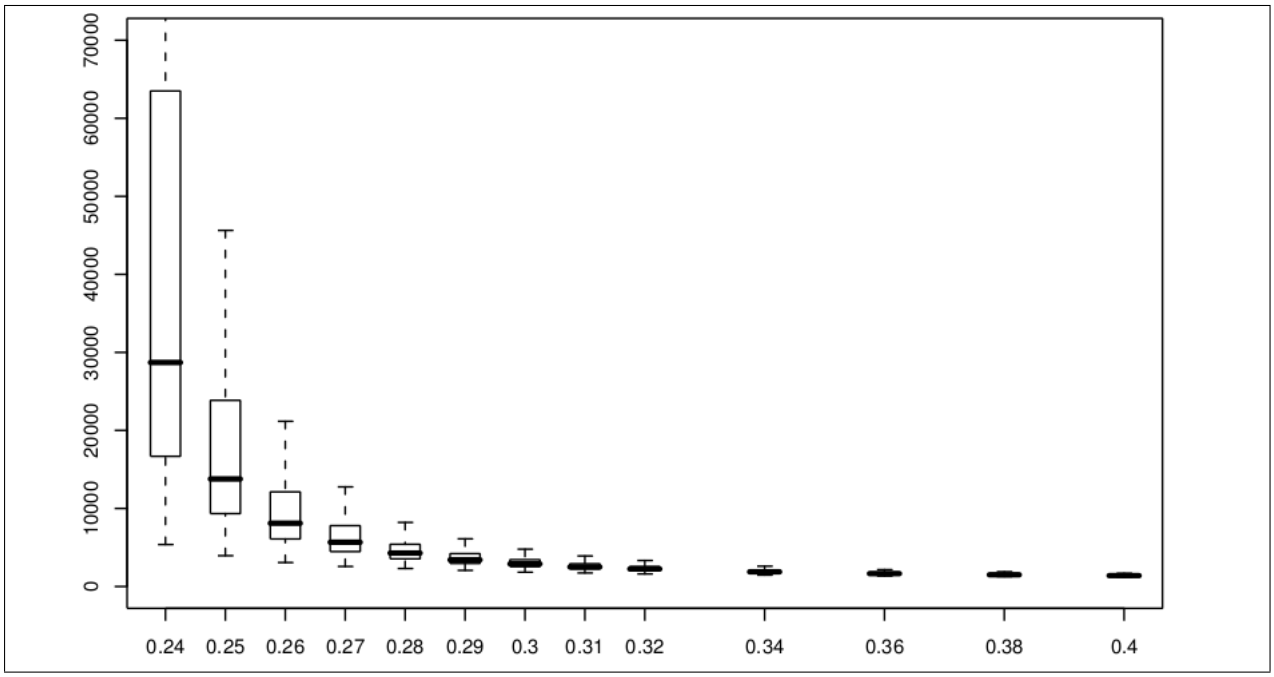

Fig. 1: Boxplot for total number of keys examined by Heninger and Shacham algorithm for $2048 - bit$ N, varying δ. This figure has been taken from [HS09] and is given here to show that our results closely match those of [HS09] as also suggested by our theoretical analysis.

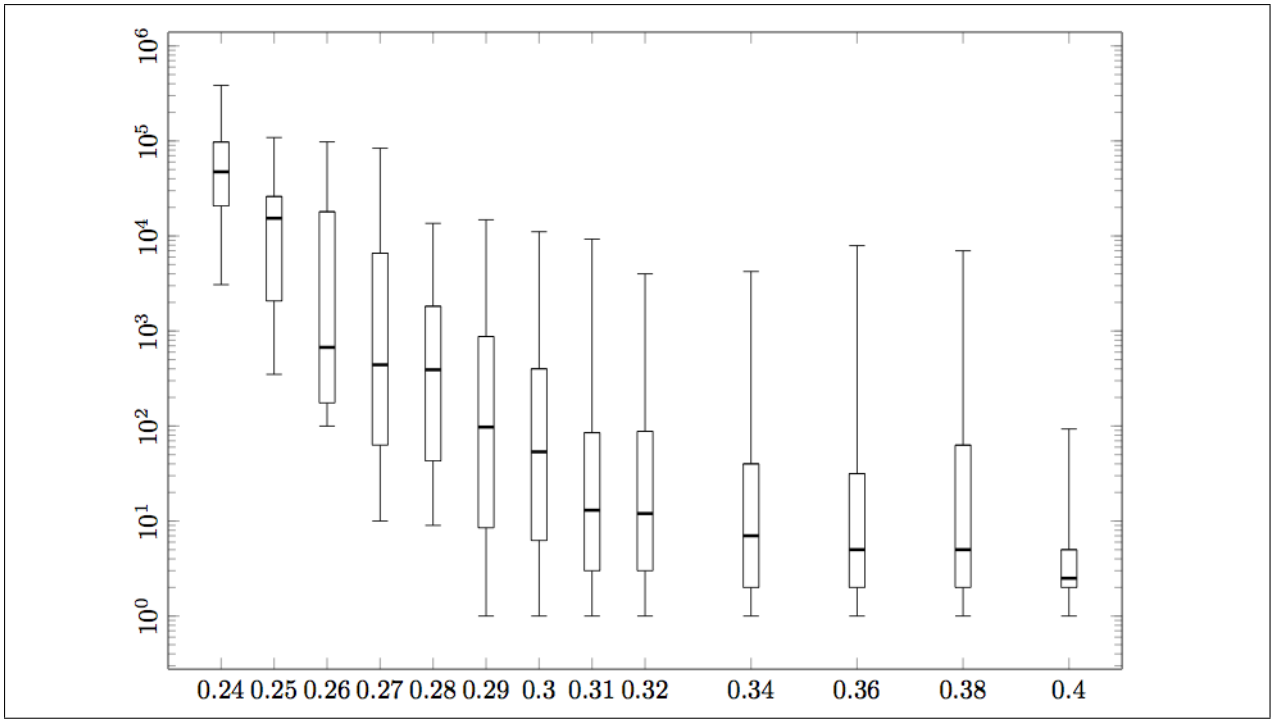

Fig. 2: Boxplot for total number of keys examined by our algorithm for $2048 - bit$ N, varying δ

[HSH+09] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: cold-boot attacks on encryption keys. volume 52, pages 91–98, New York, NY, USA, May 2009. ACM.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[MSSG10] Subhamoy Maitra, Santanu Sarkar, and Sourav Sen Gupta. Factoring RSA modulus using prime reconstruction from random known bits. In *Proceedings of the Third international conference on Cryptology in Africa*, AFRICACRYPT'10, pages 82–99, Berlin, Heidelberg, 2010. Springer-Verlag.

[PPS12] Kenneth G. Paterson, Antigoni Polychroniadou, and Dale L. Sibborn. A coding-theoretic approach to recovering noisy RSA keys. In *Proceedings of the 18th international conference on The Theory and Application of Cryptology and Information Security*, ASIACRYPT'12, pages 386–403, Berlin, Heidelberg, 2012. Springer-Verlag.

[QC82] J.-J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronics Letters*, 18(21):905–907, 1982.

[RS86] Ron L. Rivest and Adi Shamir. Efficient factoring based on partial information. In *Proceedings of a workshop on the theory and application of cryptographic techniques on Advances in cryptology—EUROCRYPT '85*, pages 31–34, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[SGM11] Santanu Sarkar, Sourav Sen Gupta, and Subhamoy Maitra. Reconstruction and error correction of RSA secret parameters from the msb side. In *WCC 2011 - Workshop on coding and cryptography (2011)*, pages 7–16, 2011.

[SGM12] Santanu Sarkar, Sourav Sen Gupta, and Subhamoy Maitra. Reconstruction and error correction of RSA private keys from the msb side. Technical report, Indian Statistical Institute, Kolkata, India, July 2012.

[SM12] Santanu Sarkar and Subhamoy Maitra. Side channel attack to actual cryptanalysis: Breaking CRT-RSA with low weight decryption exponents. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 476–493. Springer Berlin Heidelberg, 2012.

A Simulation of our algorithm for factoring 6-bit numbers.

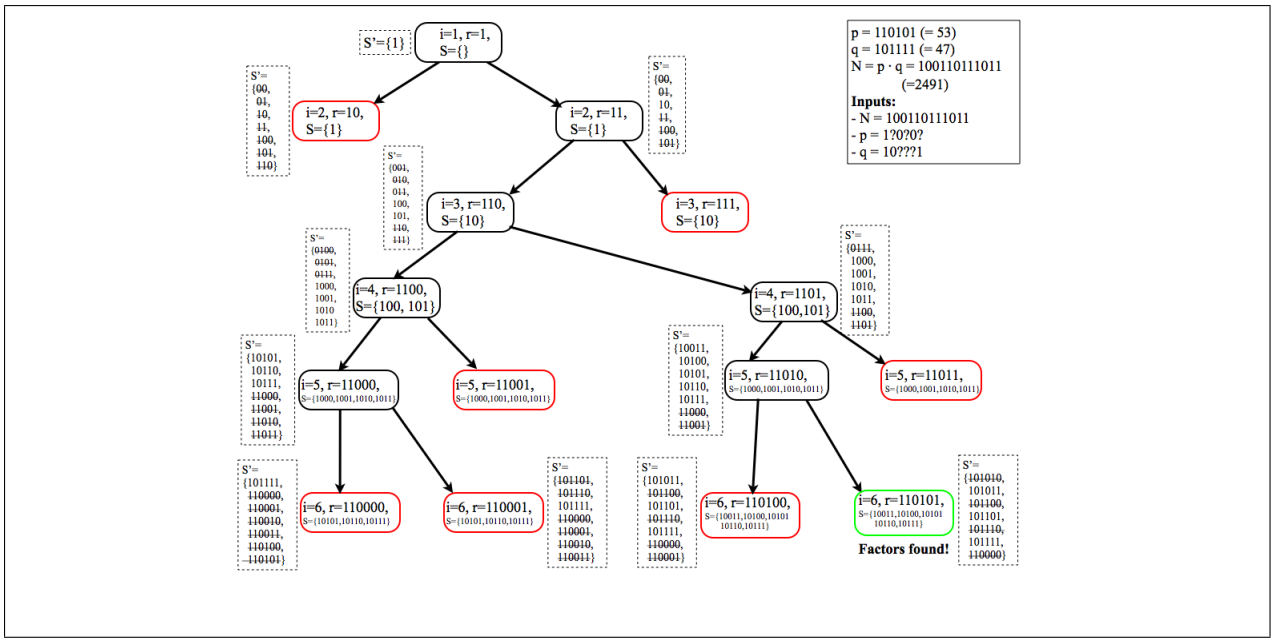

Fig. 3: Run of our algorithm for 6-bit numbers.