Side Channel Attack to Actual Cryptanalysis: Breaking CRT-RSA with Low Weight Decryption Exponents

Santanu Sarkar and Subhamoy Maitra

Applied Statistics Unit, Indian Statistical Institute,
203 B. T. Road, Kolkata 700 108, India
`sarkar.santanu.bir@gmail.com, subho@isical.ac.in`

Abstract. Towards the cold boot attack (a kind of side channel attack), the problems of reconstructing RSA parameters when (i) certain bits are unknown (Heninger and Shacham, Crypto 2009) and (ii) the bits are available but with some error probability (Henecka, May and Meurer, Crypto 2010) have been considered very recently. In this paper we exploit the error correction heuristic proposed by Henecka et al to show that CRT-RSA schemes having low Hamming weight decryption exponents are insecure given small encryption exponents (e.g., $e = 2^{16} + 1$). In particular, we show that the CRT-RSA schemes presented by Lim and Lee (SAC 1996) and Galbraith, Heneghan and McKee (ACISP 2005) with low weight decryption exponents can be broken in a few minutes in certain cases. Further, the scheme of Maitra and Sarkar (CT-RSA 2010), where the decryption exponents are not of low weight but they have large low weight factors, can also be cryptanalysed. We also identify a few modifications of the error correction strategy that provides significantly improved experimental outcome towards the cold boot attack.

Keywords: Cold Boot Attack, CRT-RSA, Cryptanalysis, Error Correction, Exponents, Hamming Weight, RSA.

1 Introduction

Side channel attack. Side channel cryptanalysis is now a quite popular technique for evaluating cryptographic schemes and this method usually considers additional information available from the physical implementation of a cryptosystem, rather than exploiting the theoretical weaknesses of the algorithm itself. The additional information may be obtained from timing information, power consumption, electromagnetic leaks etc. and the attack may very well exploit technical knowledge of the internal operation of the system on which the algorithm is implemented. The initial research in this area is pioneered by Kocher [18].

Recently, the idea of cold-boot attack has been presented in [12] that shows it is possible to exploit degraded data from the computer memory to attack cryptosystems such as DES, AES, RSA etc. This idea has been studied in more detail in [14] that shows that if certain percentage of bits of the RSA secret key are available, then it is possible to reconstruct the complete secret key (or in other words, it is possible to factorize the RSA modulus). Subsequently, in [13], a model has been considered, where the bits of the secret key are available with some probability of error. In this paper we study the work of [13] in more detail.

In general, the side channel attacks use the existing cryptanalytic techniques with additional (side channel) information. In contrast, in this paper we exploit the algorithm developed for side channel attacks [13], that is applied for a direct attack on certain versions of CRT-RSA with no extra hints or other information. Further, we also provide certain modifications on the algorithm of [13] to (heuristically) improve the results of [13]. This improved strategy can immediately be used for the side channel cryptanalysis presented in [13] which is related to cold-boot attack [12].

We also like to refer the recent paper [6] related to noisy factoring where new attacks have been proposed whose running time is essentially the "square root" of exhaustive search. In this case [6, Section 4], that attack considers that the noisy version of one of the RSA primes is available. However, the cold-boot attack model that we consider here, is different from [6] as the noisy versions of more than one secret parameters of RSA variants are in the hand of cryptanalyst.

RSA. In the seventies, the path-breaking idea of public key cryptosystem has been introduced by Diffe and Hellman [10] and as an outstanding follow-up, RSA public key cryptosystem [27] has been proposed by Rivest, Shamir and Adleman. RSA is undoubtedly the most attractive research area in cryptology with immediate applications in practice.

The RSA cryptosystem and several variants of it are in use for applications related to secure data exchange mechanisms. The encryption as well as the decryption process in RSA use modular exponentiation. As square and multiply is the most popular method for modular exponentiation, it is immediate to note that the cost is low for small exponents.

Before proceeding further, let us briefly explain the RSA public key cryptosystem. In RSA, a large integer N is generated such that $N = pq$, where p, q are primes of same bit lengths. The encryption and decryption exponents are denoted by e, d respectively and they are chosen in such a manner that $ed \equiv 1 \bmod \phi(N)$, where $\phi(N) = \phi(pq) = (p-1)(q-1)$, the Euler's totient function. The parameters e, N are distributed as the public key and the part d is kept secret. In the encryption process, we have $C = M^e \bmod N$, whereas, the decryption is performed as $M = C^d \bmod N$.

It is clear that the cost of modular exponentiation can be reduced if one can reduce the exponents e, d. However, $ed > \phi(N)$ provides the constraint that one cannot make both e, d small. For any integer x, let us denote its bit-length as ℓ_x and thus $\ell_e + \ell_d \geq \ell_N$. Towards making the decryption process faster, the secret decryption exponent d has to be made small. In this direction, using the idea of continued fraction, Wiener [28] showed that when $d < \frac{1}{3}N^{\frac{1}{4}}$, one can factor N

efficiently. Later, using lattice based techniques, this result has been improved by Boneh and Durfee [3, 2] till the upper bound $N^{0.292}$. To achieve further efficiency during encryption process, small e is considered. Coppersmith [7] has shown that RSA with very small e, e.g., $e = 3$ is not secure. For practical purposes, little larger encryption exponents are used. For example, it is a common practice to use $e = 2^{16} + 1$ and it is believed to be quite secure. Given small e, d becomes of the order of N, and the decryption process will be much less efficient than the encryption.

CRT-RSA. To achieve further efficiency during decryption, Wiener [28] prescribed use of Chinese Remainder Theorem (CRT) that has earlier been studied by Quisquater and Couvreur [26]. This is known as CRT-RSA. 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 [24] standard for the RSA cryptosystem, it is recommended to store the RSA secret parameters as a tuple $(p, q, d, d_p, d_q, q^{-1} \bmod p)$. For all the cryptanalytic strategy we mention here, the term $q^{-1} \bmod p$ could not be exploited. Thus, in this paper, we will refer the *secret key* of RSA as a tuple $SK = (p, q, d, d_p, d_q)$. Let us now discuss the cryptanalytic results related to CRT-RSA. The birthday attack has been pointed out by Pinch (as referred in [25]) in case of very small d_p, d_q (one may also have a look at [23]). Further, if $d_p, d_q < N^{0.073}$, one can factor N in polynomial time [17]. In [16], it has been shown that CRT-RSA is weak if $d_p - d_q$ is known and d_p, d_q are smaller than $N^{0.099}$. Broadly speaking, it is easy to see that CRT-RSA can be broken in $O(e)$ time if $d_p - d_q$ can be obtained with small effort.

There are also some important results related to RSA variants under the fault attack. Boneh et al [4] showed that CRT-RSA implementations are vulnerable in this regard. Later Coron et al [8, 9] extended the results of [4]. Recently, Brier et al [5] have presented alternative key-recovery attacks on CRT-RSA signatures under fault model.

RSA and CRT-RSA variants. There are several proposals on RSA and CRT-RSA key generation algorithms such that e is small and the secret parameters have certain special structures. For example, Lenstra [19] pointed out that by taking N with half of most significant bits to be zero, one would obtain around 30% advantage in encryption and decryption process. Similar idea for using large number of zeros in the binary representation of decryption exponent has also been used in several papers. In such a case, the multiplication effort will be reduced a lot in square and multiply algorithm. Initially Lim and Lee [21] considered the RSA keys with relatively low Hamming weight of the decryption exponent d. Later, Galbraith et al [11] proposed a key generation algorithm for CRT-RSA. Using that idea, one can generate CRT-RSA modulus N which allows the cost of encryption and decryption to be balanced according to the requirements of the applications. For faster decryption, one can choose d_p, d_q with low Hamming weight. In this regard, Galbraith et al [11] mentioned

"In some settings we may also want to choose the d_i to have low Hamming weight. This is easily done if the k_i are small."

Towards the security analysis, for small e, the estimated time complexity to attack such a scheme [11] has been presented as $O\left(\sqrt{w_{d_p}}\binom{\ell_{d_p}/2}{w_{d_p}/2}\right)$, where w_x is the Hamming weight of the binary representation of the integer x. In line of the work of [11], another efficient scheme has been proposed in [22] that also relied on large low weight factors in the decryption exponent d_p, d_q. The security analysis of [22] show that the exhaustive search for the low Hamming weight factors in the decryption exponents is an approach to attack such a scheme. Note that the schemes of [21, 11, 22] are motivated towards implementing on low end devices with limited computational power (such as smart card).

Our Contribution. To the best of our knowledge, there has been no cryptanalytic result on the security of schemes [21, 11, 22] so far. For the first time, in contrary to the claims in [21, 11, 22], we show that the ideas exploiting low weight integers in the secret decryption exponents, can be broken much faster. The basic technique we use in this paper is the work of [13] related to error correction of RSA secret key. In Crypto 2010, Henecka et al [13] studied the case when the bits of SK were known with some error probability for each bit. We refer a noisy version of SK as $\tilde{SK}$, i.e., $\tilde{SK} = (\tilde{p}, \tilde{q}, \tilde{d}, \tilde{d_p}, \tilde{d_q})$. That is, each bit of the parameters in SK is considered to be flipped with some probability $\delta \in [0, \frac{1}{2})$. The authors [13] could show that one can correct the errors in the secret key (i.e. recover the secret key) in polynomial time (for small e) when the error rate δ is less than $0.237, 0.160, 0.084$ when noisy versions of (p, q, d, d_p, d_q) or (p, q, d) or (p, q) are available.

The algorithm presented in [13] guesses the bits of one of the primes and then uses the reconstruction technique for cold-boot attack in [14] as to get approximations of the other parameters in SK. The verification of each guess is achieved by comparing the Hamming distance of the guess with the erroneous version of SK obtained through side-channel attacks. This is equivalent to pruning the search space towards the correct solution, and hence higher bit-error can be corrected if one uses more parameters from SK during the pruning phase.

In CRT-RSA situation, we have $ed_p \equiv 1 \bmod (p-1)$. Thus, one can write $ed_p = 1 + k_p(p-1)$ where $k_p < e$. Similarly we have $ed_q = 1 + k_q(q-1)$ where $k_q < e$. For small values of e, one may assume k_p, k_q are known to the attacker in $O(e)$ time complexity as we explain in Section 2. In general, for a randomly chosen integer x, we have $w_x \approx \frac{\ell_x}{2}$. However, for efficient decryption, sometimes w_{d_p}, w_{d_q} are taken significantly smaller than the random case. For example, consider that $\ell_{d_p} = \ell_{d_q} = 512$ and $w_{d_p}, w_{d_q} \approx 50$. In such a scenario, one can take the all zero bit string as error-incorporated (noisy) presentation of d_p, d_q, where the error rate is around $\frac{50}{512} \approx 10\%$. As the error rate is significantly small, one can apply the error correcting algorithm of [13] to recover the secret key. Denoting the time complexity of the error-correction algorithm [13] as τ, our strategy attacks the schemes [21, 11] in $\tau O(e)$ time, and the scheme [22] in $\tau O(e^3)$ time.

While attacking the schemes [21, 11, 22], one can attempt to recover all the bits of p as it is done for the error correcting algorithm in [13]. However, one can also try to construct only the least significant half of p using the same strategy and then use the lattice based result of [1] to get the complete p. While describing the experimental results, we present separate data for constructing all the bits of p and only least significant half of p.

While applying the heuristic [13], we noted a few modifications that can improve the performance significantly and the central idea is as follows. Instead of a single fixed threshold related to bit-matching in [13], we use multiple thresholds towards the motivation that we involve several constraints on the secret parameters in our case whereas a single constraint has been taken into consideration in [13].

Parameters	Upper bound of δ [13]		Success probability (expt.)		upper bound of δ
	theoretical	experimental	[13]	our	achieved in our expt.
(p, q)	0.084	0.08	0.22	0.61	0.12
(p, q, d)	0.160	0.14	0.15	0.52	0.17
(p, q, d, d_p, d_q)	0.237	0.20	0.21	0.50	0.25

Table 1. Experimental results of [13] with maximum possible δ as available from [13, Section 6, Tables 2, 3, 4].

To present a glimpse of our improvement, let us provide a brief comparison of our results with that of [13] in Table 1. For the experiments, we only refer to the results at maximum value of δ in [13] (and show that our success probability is better at that point) because our main contribution is to show that we can go significantly beyond the bound of δ in [13] with our heuristic strategy in Algorithm 2 (Section 3). See Section 4 for the detailed experimental results. For any other results describing lower error rates presented in [13], we always obtain improved success probability and those are not explicitly mentioned here.

We like to point out that apart from the specific attack on CRT-RSA with certain parameters, our improved heuristic can correct more noise than the existing strategy [13] for cold-boot attack in general.

Roadmap. In Section 2, we efficiently exploit the error correction strategy of [13] to show that CRT-RSA schemes that involve low weight secret parameters are not secure. We point out that the CRT-RSA based schemes of [21, 11] with certain parameters can be broken in a few minutes (Section 2.1) and also present cryptanalytic results on the scheme of [22] (Section 2.2). Further, in the process, we provide modifications to the error correction heuristic of [13] that provides significantly improved experimental results. This is presented in

Section 3. Detailed experimental results are presented in Section 4. Conclusion of this paper is presented in Section 5.

2 The idea of Cryptanalysis

We start with the basic relations of CRT-RSA, such as $ed_p = 1 + k_p(p-1) \Rightarrow k_p - 1 \equiv k_p p \bmod e$, and $ed_q = 1 + k_p(q-1) \Rightarrow k_q - 1 \equiv k_q q \bmod e$. From these we get $k_q \equiv (k_p - 1)\left(k_p(1-N) - 1\right)^{-1} \bmod e$. Since both $k_p, k_q < e$ and from the above equation a choice of k_p fixes k_q, there are $O(e)$ possible choices for the pairs (k_p, k_q). As we have assumed, the Hamming weight of d_p, d_q are considered to be significantly lower than the random case. Further, the presence of 1's in the binary representation of d_p, d_q are considered to be i.i.d. For better explanation, from now on, we will assume that k_p, k_q are known to the attacker and finally the complexity of the attack will be obtained by multiplying an $O(e)$ factor, unless mentioned otherwise.

Our idea is to guess a few bits (say a many bits) of p from the least significant side and let the corresponding integer be p'. From p', we get an approximation q' of q. From p', q' and using the knowledge of e, k_p, k_q, we obtain the approximations of d_p, d_q, that we denote by d'_p, d'_q respectively. If the Hamming weights of d'_p, d'_q are less than some predefined threshold, then p' would be a possible choice of p. This process will be repeated until we have obtained a set A of possible guesses p' for p. Then we extend the solutions by adding a more bits in the more significant side with the possible partial solutions in A. The process continues till we get a set of possible solutions for p itself.

```
  Input: $N, e, k_p, k_q$ and $a, C$
  Output: Set $A$, containing possible guesses for $p$.
1 Initialize $b = 0, A = \emptyset, A_{-1} = \{\lambda\}, i = 1$;
2 while $b < \frac{\ell_N}{2}$ do
3     $A = \{0,1\}^a \| A_{-1}$;
4     For each possible options $p' \in A$, calculate $q' = (p')^{-1} N \bmod 2^{b+a}$;
5     For each $p', q'$, calculate
      $d'_p = (1 + k_p(p'-1))\, e^{-1} \bmod 2^{b+a}, d'_q = (1 + k_q(q'-1))\, e^{-1} \bmod 2^{b+a}$;
6     If the number of 0's taking together the binary patterns of $d'_p, d'_q$ in the
      positions $b$ to $b+a-1$ from the least significant side is less than $C$, then
      delete $p'$ from $A$;
7     If $b \neq 0$ and $A = \emptyset$, then terminate the algorithm and report failure;
8     $A_{-1} = A$; $b = b + a$; $i = i + 1$;
  end
9 Report $A$;
```

Algorithm 1: Reconstruction algorithm for p.

In Algorithm 1, we present the algorithm formally. For that we need to use certain notations. Given two binary strings u_1, u_2, by $u_1 \| u_2$ we mean the concatenation of the strings. With abuse of notation, for two integers x, y, by $x \| y$ we denote the integer formed by the concatenation of the binary representations of x, y. We also consider the notation $X \| Y = \{x \| y : x \in X, y \in Y\}$. By λ we mean an empty or null string. Steps 4 and 5 in Algorithm 1 can be calculated efficiently using the relations [13, Equations (8), (10), (11)].

2.1 Cryptanalysis of [21, 11]

In [21, 11], CRT-RSA secret keys are generated in a manner such that the weights of d_p, d_q are small. By δ we denote the probability that a bit of d_p or d_q is 1. Thus, δ can be estimated as $\frac{w_{d_p}}{\ell_{d_p}}$ or $\frac{w_{d_q}}{\ell_{d_q}}$. Following the theoretical results of [13], we immediately get the result as below. For the sake of completeness, a detailed analysis regarding this is available in Appendix A.

Theorem 1. *Let $a = \lceil \frac{\ln \ell_N}{4\epsilon^2} \rceil, \gamma_0 = \sqrt{(1 + \frac{1}{a}) \frac{\ln 2}{4}}$ and $C = a + 2a\gamma_0$. We also consider that the parameters k_p, k_q of CRT-RSA are known. Then one can obtain p in time $O(l_N^{2+\frac{\ln 2}{2\epsilon^2}})$ with success probability greater than $1 - \frac{2\epsilon^2}{\ln \ell_N} - \frac{1}{\ell_N}$ if $\delta \le \frac{1}{2} - \gamma_0 - \epsilon$.*

To maximize δ, we need that ϵ should converge to zero and in such a case a tends to infinity. Then the value of γ_0 converges to 0.416. Thus, asymptotically Algorithm 1 works when δ is less than $0.5 - 0.416 = 0.084$. However since in this case a becomes very large, the algorithm will not be efficient and may not be implemented in practice. This is the reason, experimental results could not reach the theoretical bounds in [13].

Generally, d_p, d_q are taken to be of same bit size which is equal to $\frac{\ell_N}{2}$. Thus, following the idea of Theorem 1 above, one can cryptanalyze CRT-RSA having $w_{d_p}, w_{d_q} \le 0.04\ell_N$ in $O(e \cdot \text{poly}(\ell_N))$ time. For each possible option of k_p, k_q (this requires $O(e)$ time), one needs to apply Algorithm 1 to obtain p. It is indeed clear that for small e the attack remains efficient.

In [21, Page 9, end of paragraph 3], example parameters have been proposed, where $\ell_N = 768$, $\ell_{d_p} = 384$ and $w_{d_p} = 30$. This falls under the condition mentioned above and we could cryptanalyze all the CRT-RSA keys with such parameters in a few minutes in practice. In another example [21, Table 2, Section 7], it has been considered that $\ell_N = 768$, $\ell_{d_p} = 377$ and $w_{d_p} = 45$ and $e = 257$. In this case $\delta = \frac{w_{d_p}}{\ell_{d_p}} \approx 0.12 > 0.08$, and thus it is not in the bound given in Theorem 1 and so Algorithm 1 would not work as it is. However, in the next section (Section 3) we will present some modifications over Algorithm 1 to get Algorithm 2 that provides significantly improved results experimentally than what presented in Theorem 1. That helps us in easily breaking CRT-RSA with the above mentioned parameters in a few minutes again.

In [11, Figure 1], parameters are proposed as $(\ell_e, \ell_{d_p}, \ell_{k_p}) = (176, 338, 2)$ with $w_{d_p} = 38$. Note that in this situation, one could easily obtain k_p and k_q, even without trying $O(e)$ steps. Here $\delta = \frac{38}{338} \approx 0.11$. Using Algorithm 2 discussed in Section 3, one can break the CRT-RSA scheme with such parameters mentioned in [11] within a few minutes.

2.2 Cryptanalysis of [22]

In [22], the CRT-RSA decryption exponents d_p, d_q have been chosen in a slightly different manner. Here the weight of d_p, d_q are not small, but they are of the form $d_p = d_{p_1} d_{p_2}$ and $d_q = d_{q_1} d_{q_2}$. The factors d_{p_1} and d_{q_1} are of size $O(e)$ and the other factors d_{p_2} of d_p and d_{q_2} of d_q are of significantly small Hamming weight. So, in this case we have $ed_{p_1} d_{p_2} = 1 + k_p(p-1)$ and $ed_{q_1} d_{q_2} = 1 + k_q(q-1)$. There are $O(e^2)$ choices of d_{p_1}, d_{q_1}. Since $k_q \equiv (k_p - 1)\left(k_p(1-N) - 1\right)^{-1} \bmod e$, within $O(e^3)$ many attempts one can get the correct choice of $(d_{p_1}, d_{q_1}, k_p, k_q)$. Hence one can consider that the attacker knows d_{p_1}, d_{q_1}, k_p and k_q, perform the attack and then multiply the effort by $O(e^3)$ to get the total time complexity of the cryptanalysis. Let a many LSBs of p be known and p' be the corresponding integer. Then a many LSBs of d_{p_2}, d_{q_2} can be obtained through the following identities:

$$\begin{aligned} d_{p_2} &\equiv (ed_{p_1})^{-1}(1 + k_p(p'-1)) \bmod 2^a, \\ d_{q_2} &\equiv (ed_{q_1})^{-1}\left(1 + k_q\left(N(p')^{-1} \bmod 2^a - 1\right)\right) \bmod 2^a. \end{aligned} \tag{1}$$

We use the Equation (1) in step 5 of the Algorithm 1 instead of what is given there towards the cryptanalysis of [22]. Here $\ell_{d_{p_2}} \approx \frac{\ell_N}{2} - \ell_e$. When $w_{d_{p_2}}, w_{d_{q_2}} \leq 0.08\left(\frac{\ell_N}{2} - \ell_e\right)$, one could cryptanalyze the CRT-RSA scheme with the parameters proposed in [22] in time $O(e^3 \ell_N^{2+\frac{\ln 2}{2\epsilon^2}})$.

Five challenges have been presented in [22], where $e = 2^{16}+1$, $\ell_N = 1024$ and $w_{d_{p_2}} = w_{d_{q_2}} = 40$. As both d_{p_1} and d_{q_1} are of $O(e)$, $\ell_{d_{p_2}} \approx \ell_{d_{q_2}} \approx 512 - 16 = 496$. Hence $\delta = \frac{40}{496} \approx 0.08$. Thus, the proposal of [22] can be cryptanalysed using Algorithm 1 with a modification in step 5 as described above.

Let us now explain the efficiency of our cryptanalysis on the proposal of [22] as we could not break it in real time as had been done on the examples of [21, 11]. Note that for e as described in [22], e^3 is around 2^{48} and that many runs of Algorithm 1 or Algorithm 2 are required.

The parameters of [22] are so chosen that if one tries to go for an exhaustive search, then it will require around 2^{94} effort. Based on this, it has been claimed [22] that such a scheme is secure as the best possible factorization strategy using NFS [20] requires around 2^{86} time complexity and one cannot attack the scheme in a lower complexity than that.

Now we show that one can indeed attack the scheme of [22] in a time complexity much less than 2^{86}. One can implement the attack with all possible values of $d_{p_1}, d_{q_1}, k_p, k_q$ that requires 2^{48} many invocations of Algorithm 1 or Algorithm 2 when e is 16-bit integer. As Algorithm 2 works significantly better than Algorithm 1, we estimate the time complexity of each invocation of Algorithm 2. Given a block length $a = 10$, to get p, we need $\lceil \frac{512}{10} \rceil = 52$ many generations of set A, where we bound the number of solutions in the set A by 1000 in the experiments. We can estimate the time complexity for each invocation of Algorithm 2 with the above parameters as $52 \cdot 1000 < 2^{16}$. Hence, the total complexity of the attack is around $2^{48+16} = 2^{64}$, which is significantly smaller than 2^{86}.

While the idea presented in [22] can be cryptanalyzed, it cannot be broken in a few minutes experimentally as it could be done for [21, 11]. One may actually explore the ideas of countermeasure from [22]. To protect/blind the secret exponents, one needs to use the product of one large integer (of small weight) and one very small integer such that the weight of the product is not small (please see Section 2.2). Proper choice of parameters (in particular bit length of the smaller and larger factors) will make the attack less effective. As an example, instead of choosing 16-bit k_p, k_q, one may try larger ones (say, 32 bits). This will increase d_{p_1}, d_{q_1} to 32 bits. In this case, the attacker needs to know d_{p_1}, d_{q_1}, k_p and the attack complexity will this increase.

3 Heuristics for further improvement of the error correction algorithm [13]

The theoretical bounds on the noise rate δ in [13] for which SK can be recovered from a noisy version of it are as follows: (i) $\delta < 0.237$, when $\tilde{p}, \tilde{q}, \tilde{d}, \tilde{d_p}, \tilde{d_q}$ are available, (ii) $\delta < 0.160$, when $\tilde{p}, \tilde{q}, \tilde{d}$ are available, and (iii) $\delta < 0.084$ when $\tilde{p}, \tilde{q}$ are available. However one cannot achieve these bounds due to high length of the block size. The experimental bounds achieved in [13, Section 6, Tables 2, 3, 4] are presented as 0.20, 0.14 and 0.08 for cases 1, 2 and 3 respectively with success probability less than 0.25. In this section we explain certain heuristics to improve these experimental results significantly. We present experimental results having error rates higher than the theoretical upper bound of [13]. This we get for the parameter $a = 10$ which is much lower than the values used in the experiments of [13] and increasing a in our strategy improves our results further. Let us now present the broad ideas behind our improvements.

Different values of the threshold C. Instead of one fixed threshold C, we take different thresholds in different steps, that depend on the value $b+a$. During the pruning, we count the number of bits at which the noisy parameters and the possible solutions match for the positions 0 to $b + a - 1$. Thus the number of comparison for each parameter is $(b+a)$ at each step and then based on that we decide whether we will accept or reject a solution. Thus we consider a cumulative measure, where for the initial steps, the bit strings compared are of the lesser size and as the solution grows, the bit strings in comparison are of larger size. The threshold parameters are chosen based on the error rate and length of the-then solutions (which are actually $b + a$ at that point of time).

Multiple constraints on each round. Instead of considering the total number of mismatched bits for each component of the secret key, all possible constraints are considered at the same time in our strategy. Suppose we want to factor

$N = pq$ where p, q are available with some noise. Consider an instance of the algorithm where we have reached up to the bit position i.

For proceeding further, let us have a few notations. For an integer x, by $x[i]$, we denote the i-th least significant bit of x. Further, by $x_{[i]}$, we mean the bit-string $x[i], x[i-1], \ldots, x[1], x[0]$ and this will also be interpreted as an integer. As an example, for a prime, say, $p = 23$, we have the binary representation 10111 and thus, $p_{[3]} = 0111$, which is 7 as an integer.

Let the number of matched bits between the partial solution p' and the corresponding bits of the noisy version $\tilde{p}$, i.e., $\tilde{p}_{[i]}$ be μ_1 and for q' and $\tilde{q}_{[i]}$ be μ_2. We impose constraints on both the values of μ_1, μ_2 along with $\mu_3 = \mu_1 + \mu_2$ to achieve better pruning. In [13], the pruning was applied on the sum of the individual values only. Here, in case of m many components of the secret key, total number of constraints in each iteration would be $\sum_{i=1}^{m} \binom{m}{i} = 2^m - 1$. When we work with the five parameters p, q, d, d_p, d_q, we use a total of 31 constraints instead of 1 as mentioned in [13].

Suppose that the secret key has m components. For the l-th component $(1 \leq l \leq m)$ of the key, let $\mu_{2^{l-1}}$ denote the number of matched bits between the partial solution and the noisy version of the component at the corresponding bits. Then for each individual component, numbers of matched bits are given by $\{\mu_1, \mu_2, \mu_4, \ldots, \mu_{2^{m-1}}\}$ respectively. Now, for any general $k \in [1, 2^m - 1]$, we define the term μ_k as follows: $\mu_k = \sum_{i=1}^{m} k_i \mu_{2^{i-1}}$, where k_i are the bits of k for $1 \leq i \leq m$. Thus the total number of matched bits in all components of the secret key is given by $\mu_{2^m-1} = \sum_{i=1}^{m} \mu_{2^{i-1}}$. In practice, when we work with $m = 5$ parameters p, q, d, d_p, d_q of the secret key, $\mu_{31} = \mu_1 + \mu_2 + \mu_4 + \mu_8 + \mu_{16}$ represents the cumulative sum of matched bits between the partial solution and corresponding bits of the noisy version for all parameters.

Value of threshold parameters and its run-time modifications. For each μ_i, we choose the value of the threshold C_i^{a+b} depending upon the noise rate δ and the value of $a + b$ at that stage. For a fixed noise rate δ, we choose the minimum C_i^{a+b} such that

$$\sum_{j=1}^{C_i^{a+b}} \binom{w_i(a+b)}{j} \delta^j (1-\delta)^{a+b-j} > \nu, \qquad (2)$$

where w_i is the Hamming weight of i as we have mentioned earlier. In the experiments we take $\nu = 0.99$ for $a + b < 150$ and $\nu = 0.98$ for $a + b \geq 150$. We choose the thresholds like this so that the possibility of rejecting a correct partial solution remains very low. It is clear that one cannot allow the size of A to increase exponentially. Thus one needs to keep some upper bound on $|A|$ while running the algorithm. Let $|A|$ be restricted by a constant upper bound B. Consider m secret parameters and take certain threshold for each μ_i. While creating the set A from A_{-1} in a loop, if $|A| > B$, we reduce $C_{2^m-1}^{a+b}$ by 1. One may ask, why do we reduce only the threshold corresponding to $\mu_{2^m-1} = \mu_1 + \mu_2 + \ldots + \mu_{2^{m-1}}$. We have tried in manipulating other thresholds as well, but found that this is quite an effective idea to obtain good experimental results. The study of such thresholds and their modifications during the run of the algorithm is an interesting question and requires serious attention that is not in the scope of this work.

The modified algorithm. Based on the above discussion, our improved error correction strategy is presented in Algorithm 2. We have presented the algorithm with all the five parameters $\tilde{p}, \tilde{q}, \tilde{d}, \tilde{d_p}, \tilde{d_q}$, though one can easily modify it for less number of parameters.

```
Input: $N, e, k, k_p, k_q$
Input: $\tilde{p}, \tilde{q}, \tilde{d}, \tilde{d_p}, \tilde{d_q}$
Input: $a, B$ and threshold parameters as described in (2).
Output: Set $A$, containing possible guesses for $p$.
1  Initialize $b = 0, A = \emptyset, A_{-1} = \emptyset$;
2  while $b < \frac{\ell_N}{2}$ do
3      $A = \{0,1\}^a \| A_{-1}$;
4      For each possible options $p' \in A$, calculate $q' = (p')^{-1} N \bmod 2^{b+a}$;
5      Calculate $d' = (1 + k\,(N + 1 - p' - q'))\,e^{-1}) \bmod 2^{b+a}$;
6      Calculate
       $d'_p = (1 + k_p(p' - 1))\,e^{-1} \bmod 2^{b+a}, d'_q = (1 + k_q(q' - 1))\,e^{-1} \bmod 2^{b+a}$;
7      Calculate $\mu_i$'s for $i = 1$ to 31 comparing least significant $b + a$ bits of the
       noisy strings and the corresponding possible partial solution strings of
       length $b + a$, i.e., through the positions 0 to $b + a - 1$;
8      If $\mu_i < C_i^{a+b}$ for any $i \in [1, \dots, 31]$, delete the solution from $A$;
9      If $|A| > B$, reduce $C_{31}^{a+b}$ by 1 and go to Step 8;
10     If $b \neq 0$ and $A = \emptyset$, then terminate the algorithm and report failure;
11     $A_{-1} = A$; $b = b + a$;
   end
12 Report $A$;
```

Algorithm 2: Improved Error Correction algorithm.

In the next section we present experimental results to highlight the significant improvements over [13]. Theoretical analysis of Algorithm 2 (possibly exploiting statistical techniques) is indeed of interest, though it is not attempted in this initiative.

4 Experimental Results

We have implemented Algorithm 2 using C programming language (with GMP library for processing large integers) on Linux Ubuntu 2.6. The hardware platform is an HP Z800 workstation with 3GHz Intel(R) Xeon(R) CPU. Our implementation is not optimized and for each run the time required varied from a few seconds to a few minutes depending on the error rates. The time estimations presented in our tables are the averages of the time required in the successful runs only. In all the experiments, we take 1024-bit RSA with public exponent $e = 2^{16}+1$. We consider $a = 10$ and $B = 1000$. For each experiment, we generate 20 different RSA secret keys and for each secret key, we generate 20 many noisy versions by incorporating independently and uniformly distributed errors with noise rate δ. So, we have a total of 400 samples.

First we go for only two parameters. This can be interpreted in two ways: (i) the noisy versions of p, q are available with error rate δ or (ii) p, q are completely unknown, but the weights of p, q are small, i.e., $\frac{w_p}{\ell_p} \approx \frac{w_q}{\ell_q} \approx \delta$.

Note that we run Algorithm 2 till we obtain all the bits of p. However, it is known that if one obtains the least significant half of p, then it is possible to obtain the factorization of N efficiently [1]. In this case, as we need to reconstruct half of the bits of p instead of the full binary string, the success probability will increase. Keeping this in mind, in the experimental results we provide the success probability to obtain the complete bit pattern of p to compare our results with [13] as well as the success probability to obtain least significant half of p, that is actually required for the attack. We refer the second one as success probability (half) in the tables. The error rate of the order of 0.08 could be achieved with success probability 0.22 in [13], and one may see in Table 2 that our results are significantly improved.

δ	0.08	0.09	0.10	0.11	0.12	0.13
Success probability	0.61	0.36	0.19	0.06	0.02	-
Time (in seconds)	255.97	249.56	252.23	235.34	230.13	-
Success probability (half)	0.71	0.55	0.41	0.23	0.13	0.08
Time (in seconds)	68.82	66.24	66.23	66.00	60.04	61.67

Table 2. Experimental results with Algorithm 2 with two parameters p, q.

δ	0.08	0.09	0.10	0.11	0.12	0.13
Success probability	0.59	0.27	0.14	0.04	-	-
Time (in seconds)	307.00	294.81	272.72	265.66	-	-
Success probability (half)	0.68	0.49	0.25	0.18	0.08	0.02
Time (in seconds)	87.41	84.47	80.18	74.57	79.33	76.04

Table 3. Experimental results with Algorithm 2 with two parameters d_p, d_q.

In Table 3, we present experimental results related to our attack in Section 2. Taking $e = 2^{16} + 1$, we have generated CRT-RSA secret exponents d_p, d_q having small Hamming weights using the idea of [11]. In Table 3, we present experimental results taking 5-bit k_p, k_q. The results in Table 3 is slightly worse than Table 2. This is because in the least ℓ_{k_p} many bits of d_p, d_q, the error rate is not small due to the key generation algorithm of [11]. We obtained similar kinds of results for cryptanalysis of [22] too, with similar parameters as in the benchmark examples of [22, Appendix A]. However, in these cases, for practical experiments in a few minutes, we need to consider that $d_{p_1}, d_{q_1}, k_p, k_q$ are available. That is, the time need to be multiplied by 2^{48} for actual attack, when $e = 2^{16} + 1$, say.

Next, in Table 4, we consider the case with three parameters p, q, d. When $\delta = 0.14$, the success probability in [13, Table 3] has been reported as 0.15. The success probability using our modification is 0.52 in this scenario. Further we could demonstrate experimental results till $\delta = 0.17$ which is better than the theoretical bound of 0.16 in [13].

δ	0.14	0.15	0.16	0.17	0.18
Experiment	0.52	0.21	0.11	0.03	-
Time (in seconds)	461.24	430.36	412.08	407.8	-
Experiment(half)	0.69	0.48	0.23	0.14	0.07
Time (in seconds)	142.99	131.69	127.12	123.00	124.00

Table 4. Experimental results with Algorithm 2 with three parameters p, q, d.

Next we present the results will all five parameters. It is evident from Table 5 that we obtained significant improvement over the results of [13].

δ	0.20	0.21	0.22	0.23	0.24	0.25
Experiment	0.50	0.44	0.33	0.14	0.02	0.005
Time (in seconds)	699.82	639.16	607.23	580.45	540.10	502.00
Experiment(half)	0.70	0.58	0.55	0.31	0.12	0.065
Time (in seconds)	221.00	192.35	190.94	173.96	168.00	169.61

Table 5. Experimental results with Algorithm 2 with five parameters p, q, d, d_p, d_q.

Our current implementation is only towards proof-of-the-concept. The results are expected to improve further with optimized implementation. While we can work with higher error rates and the success probability of our modified version are much better than [13], we require little more running time than [13]. However, we like to point out that our improvement is not achieved at the cost of a higher running time as our results cannot be reached taking $a = 10$ by the techniques of [13]. Towards the higher error rates, the value of a in [13] varies from 20 to 29, whereas we work with a as small as 10 for all the cases. Still we achieve better success rate. Larger the size of a, larger is the set of partial solutions A. This clearly shows that our strategy performs significantly better with the new ideas of pruning where the correct solution is retained with good success rate.

We have also explored a few other implementation strategies for Algorithm 2. Let $D(x)$ be the largest integer that divides x. From the knowledge of k, k_p and

k_q, one can easily calculate $D(k), D(k_p), D(k_q)$. Thus, in the steps 5 and 6 of the Algorithm 2, one can calculate

$d' = (1 + k\,(N + 1 - p' - q'))\,e^{-1}) \bmod 2^{b+a+D(k)}$,
$d'_p = (1 + k_p(p' - 1))\,e^{-1} \bmod 2^{b+a+D(k_p)}$ and
$d'_q = (1 + k_q(q' - 1))\,e^{-1} \bmod 2^{b+a+D(k_q)}$.

Since $D(k), D(k_p)$ and $D(k_q)$ are very small in general, the improvement in terms of time complexity will not be significant and we have checked that experimentally too. In the course of optimizing the algorithm, one may note that the solution sets for these equations can be evolved rapidly by Hensel lifting. These are actually used in the time of inverse calculations in our strategy. In this case also, we did not obtain major improvements in running time.

5 Conclusion

In this paper, first we apply the recently proposed error correction strategy (motivated from cold-boot attack) for RSA secret keys [13] to actual cryptanalysis of CRT-RSA under certain conditions. We studied two kinds of schemes. The first one considers the CRT-RSA decryption keys of low weight as in [21, 11]. In these cases, we demonstrate complete break in a few minutes for 1024 bit RSA moduli. The next one considers the scenario when the decryption exponents are not of low weight, but they contain large low weight factors [22]. Though this scheme seems more resistant to our method than the ones in [21, 11], it [22] is also prone to cryptanalysis with much lower complexity than what claimed in the paper [22].

Further, we had a detailed look at the actual error correction algorithm of [13] and provided significant improvements as evident from experimental results. The experimental results are significantly better than the ones presented in [13] and more importantly, we could demonstrate that the theoretical bound of [13] can also be crossed using our heuristic. These results can directly be applied to cold-boot attack, in general, on RSA and its variants.

Acknowledgments: The authors like to thank the Centre of Excellence in Cryptology, Indian Statistical Institute for relevant support towards this research.

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, volume 1514 of Lecture Notes in Computer Science, Springer, pages 25–34, 1998.
2. D. Boneh and G. Durfee. Cryptanalysis of RSA with private key less than $N^{0.292}$. In Proceedings of Eurocrypt, volume 1592 of Lecture Notes in Computer Science, Springer, pages 1–11, 1999.
3. D. Boneh and G. Durfee. Cryptanalysis of RSA with Private Key d Less Than $N^{0.292}$. IEEE Transactions on Information Theory, 46(4):1339–1349, 2000.

4. D. Boneh, R. A. DeMillo and R. J. Lipton. On the importance of checking cryptographic protocols for faults. Journal of Cryptology, 14(2):101–119, 2001.
5. E. Brier, D. Naccache, P. Q. Nguyen and M. Tibouchi. Modulus Fault Attacks against RSA-CRT Signatures. In Proceedings of CHES 2011, volume 6917 in Lecture Notes in Computer Science, Springer, pages 192–206, 2011.
6. Y. Chen and P. Q. Nguyen. Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers. In proceedings of Eurocrypt 2012. volume 7237 in Lecture Notes in Computer Science, Springer, pages 502–519, 2012.
7. D. Coppersmith. Small Solutions to Polynomial Equations and Low Exponent Vulnerabilities. Journal of Cryptology, 10(4):223–260, 1997.
8. J.-S. Coron, A. Joux, I. Kizhvatov, D. Naccache and P. Paillier. Fault attacks on RSA signatures with partially unknown messages. In Proceedings of CHES 2009, volume 5747 in Lecture Notes in Computer Science, Springer, pages 444–456, 2009.
9. J.-S. Coron, D. Naccache and M. Tibouchi. Fault Attacks Against EMV Signatures. In Proceedings of CT-RSA 2010, volume 5985 in Lecture Notes in Computer Science, Springer, pages 208–220, 2010.
10. W. Diffie and M. E. Hellman. New directions in cryptography. IEEE Transactions on Information Theory, IT-22(6):644–654, 1976.
11. S. D. Galbraith, C. Heneghan and J. F. McKee. Tunable balancing of RSA. In Proceedings of ACISP, volume 3574 in Lecture Notes in Computer Science, Springer, pages 280–292, 2005. Available at `http://www.isg.rhul.ac.uk/~sdg/full-tunable-rsa.pdf` [last accessed December 8, 2011]
12. J. A. Halderman, S. Schoen, N. Heninger, W. Clarkson, W. Paul, J. Calandrino, A. Feldman, J. Appelbaum, and E. Felten. Lest we remember: Cold boot attacks on encryption keys. In Proceedings of USENIX Security 2008, pages 4560. USENIX, July 2008.
13. W. Henecka, A. May and A. Meurer. Correcting Errors in RSA Private Keys. In Proceedings of Crypto, volume 6223 in Lecture Notes in Computer Science, Springer, pages 351–369, 2010.
14. N. Heninger and H. Shacham. Reconstructing RSA private keys from random key bits. In Proceedings of Crypto, volume 5677 in Lecture Notes in Computer Science, Springer, pages 1–17, 2009.
15. W. Hoeffding. Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association, 58(301):13–30, 1963.
16. E. Jochemsz and A. May. A Strategy for Finding Roots of Multivariate Polynomials with new Applications in Attacking RSA Variants. In Proceedings of Asiacrypt, volume 4284 in Lecture Notes in Computer Science, Springer, pages 267–282, 2006.
17. E. Jochemsz and A. May. A polynomial time attack on RSA with private CRT-exponents smaller than $N^{0.073}$. In Proceedings of Crypto, volume 4622 in Lecture Notes in Computer Science, Springer, pages 395–411, 2007.
18. P. C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Proceedings of Crypto, volume 1109 in Lecture Notes in Computer Science, Springer, pages 104113, 1996.
19. A. Lenstra. Generating RSA Moduli with a Predetermined Portion. In Proceedings of Asiacrypt, volume 1514 in Lecture Notes in Computer Science, Springer, pages 1–10, 1998.
20. A. K. Lenstra and H. W. Lenstra, Jr., The Development of the Number Field Sieve, Springer-Verlag, 1993.

21. C. H. Lim and P. J. Lee. Sparse RSA Secret Keys and Their Generation. In Proceedings of (SAC), pages 117–131, 1996. Available at http://dasan.sejong.ac.kr/~chlim/english_pub.html [last accessed December 8, 2011]
22. S. Maitra and S. Sarkar. Efficient CRT-RSA Decryption for Small Encryption Exponents. In Proceedings of CT-RSA, volume 5985 in Lecture Notes in Computer Science, Springer, pages 26–40, 2010.
23. A. May. New RSA Vulnerabilities Using Lattice Reduction Methods. PhD thesis, University of Paderborn, Germany, 2003.
24. 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.
25. G. Qiao and K. -Y. Lam. RSA Signature Algorithm for Microcontroller Implementation. In Proceedings of CARDIS (1998), volume 1820 in Lecture Notes in Computer Science, Springer, pages 353–356, 2000.
26. J. -J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. Electronic Letters, 18:905–907, 1982.
27. 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, February 1978.
28. M. Wiener. Cryptanalysis of Short RSA Secret Exponents. IEEE Transactions on Information Theory, 36:553–558, 1990.

Appendix A

In Section 2, we have described Algorithm 1 towards cryptanalysis of CRT-RSA schemes with low weight decryption exponents when the encryption exponent is small. The theoretical result related to the algorithm has been presented in Theorem 1. Theorem 1 follows directly from the analysis of [13]. Still we explain this in detail for completeness.

We refer to Algorithm 1. Let us define a random variable X_c for the total number of 0's in the binary representation of d'_p and d'_q from the position b to $b+a-1$, where d'_p, d'_q are correct partial solution of d_p, d_q. Clearly X_c follows binomial distribution with parameters $2a$ and probability $1-\delta$. Hence

$$P(X_c = \gamma) = \binom{2a}{\gamma}(1-\delta)^{\gamma}\delta^{2a-\gamma},$$

for $\gamma = 0, \ldots, 2a$.

Now assume one can expand some incorrect partial solutions (p', q') and obtain d'_p, d'_q. Let X_b be the number of 0's in the expanded $2a$ many bits in d'_p, d'_q. To study the distribution of X_b, we consider the following heuristic assumption that every solution generated from incorrect partial solution consists of randomly chosen bits. Thus,

$$P(X_b = \gamma) = \binom{2a}{\gamma}2^{-2a}.$$

We have to choose a threshold C such that the two distributions X_c and X_b are sufficiently separated.

Take $C = a + 2a\gamma_0$ where $\gamma_0 = \sqrt{(1+\frac{1}{a})\frac{\log 2}{4}}$. Let the random variable Y_i represent the number of incorrect partial solutions that pass the threshold bound C at i-th stage. Then one can get the following result.

Lemma 1. *The expectation* $E[Y_i]$ *of* Y_i *is less than* 2^{a+1}.

Proof. Let us denote Z_g as the number of incorrect candidates from the partially correct solution and let Z_b give the count of the number of incorrect candidates from each partially incorrect solution. Thus, we have

$$\begin{aligned}
E[Y_1] &= E[Z_g], \\
E[Y_2] &= E[Z_g] + E[Z_b]E[Y_1], \\
&\vdots \\
E[Y_i] &= E[Z_g] + E[Z_b]E[Y_{i-1}] \\
&= E[Z_g] + E[Z_b]\left(E[Z_g] + E[Z_b]E[Y_{i-2}]\right) \\
&= \cdots \\
&= E[Z_g]\sum_{k=0}^{i-1} E[Z_b]^k \\
&= E[Z_g]\frac{1 - E[Z_b]^i}{1 - E[Z_b]}.
\end{aligned}$$

Now define 2^a random variables corresponding to $i = 1, \cdots, 2^a$ such that

$$Z_b^i = \begin{cases} 1 \text{ if } i\text{-th bad candidate passes the threshold} \\ 0 \text{ otherwise} \end{cases}$$

Clearly, $Z_b = \sum_{i=1}^{2^a} Z_b^i$. So,

$$\begin{aligned}
E[Z_b] &= 2^a E\left[Z_b^i\right] \\
&= 2^a Pr\left[Z_b^i = 1\right] \\
&= 2^a Pr\left[X_b \geq C\right] \\
&= 2^a Pr\left[X_b \geq 2a\left(\frac{1}{2} + \gamma_0\right)\right]
\end{aligned}$$

From Hoeffding's inequality [15] we know,

$$\begin{aligned}
Pr\left[X_b \geq 2a\left(\frac{1}{2} + \gamma_0\right)\right] \leq e^{-4a\gamma_0^2} &= e^{-4a\left(1+\frac{1}{a}\right)\frac{\log 2}{4}} \\
&= 2^{-a-1}.
\end{aligned}$$

Hence $E[Z_b] \leq 2^a \cdot 2^{-a-1} = \frac{1}{2} < 1$. So,

$$\begin{aligned} E[Y_i] &< \frac{E[Z_g]}{1 - E[Z_b]} \leq 2E[Z_g] \leq 2(2^a - 1) \text{ as } E[Z_g] \leq 2^a - 1 \\ &< 2^{a+1}. \end{aligned}$$

□

To have the time complexity of Algorithm 1 polynomial in ℓ_N, one can take, for example, $a = \lceil \frac{\ln \ell_N}{4\epsilon^2} \rceil$ and $\delta \leq \frac{1}{2} - \gamma_0 - \epsilon$. Then we get the following result.

Lemma 2. *Algorithm 1 succeeds with probability greater than* $1 - \frac{\epsilon^2}{\ell_N} - \frac{1}{\ell_N}$.

Proof. The probability of pruning the correct partial solution at one step is given by $Pr[X_c < C]$. Now

$$Pr[X_c < C] = Pr[X_c < 2a(\frac{1}{2} + \gamma_0)] \leq Pr[X_c < 2a(1 - \delta - \epsilon)] \leq e^{-4a\epsilon^2} \leq \frac{1}{\ell_N}.$$

Thus,

$$\begin{aligned} Pr[success] &= (1 - Pr[X_c < C])^{\lceil \frac{\ell_N}{2a} \rceil} \geq (1 - Pr[X_c < C])^{\frac{\ell_N}{2a}+1} \\ &\geq (1 - \frac{1}{\ell_N})^{\frac{\ell_N}{2a}+1} \geq 1 - \frac{\frac{\ell_N}{2a} + 1}{\ell_N} \geq 1 - \frac{1}{2a} - \frac{1}{\ell_N} \geq 1 - \frac{2\epsilon^2}{\ln \ell_N} - \frac{1}{\ell_N}. \end{aligned}$$

□

Using same idea of [13], it can be shown that the time complexity of our idea is $O(\ell_N^{2+\frac{\log 2}{2\epsilon^2}})$. Hence from Lemma 1 and Lemma 2, we get Theorem 1 (Section 2.1) as follows.

Theorem 1. *Let* $a = \lceil \frac{\ln \ell_N}{4\epsilon^2} \rceil, \gamma_0 = \sqrt{(1 + \frac{1}{a}) \frac{\ln 2}{4}}$ *and* $C = a + 2a\gamma_0$. *We also consider that the parameters* k_p, k_q *of CRT-RSA are known. Then one can obtain* p *in time* $O(l_N^{2+\frac{\ln 2}{2\epsilon^2}})$ *with success probability greater than* $1 - \frac{2\epsilon^2}{\ln \ell_N} - \frac{1}{\ell_N}$ *if* $\delta \leq \frac{1}{2} - \gamma_0 - \epsilon$.