RSA Cryptanalysis with Increased Bounds on the Secret Exponent using Less Lattice Dimension

Santanu Sarkar, Subhamoy Maitra and Sumanta Sarkar

Indian Statistical Institute, 203 B T Road, Kolkata 700 108, India
{santanu_r, subho, sumanta_r}@isical.ac.in

Abstract. We consider RSA with $N = pq$, $q < p < 2q$, public encryption exponent e and private decryption exponent d. Boneh and Durfee (Eurocrypt 1999, IEEE-IT 2000) used Coppersmith's method (Journal of Cryptology, 1997) to factorize N using e when $d < N^{0.292}$, the theoretical bound. Related works have also been presented by Blömer and May (CaLC 2001). However, the experimental bound for d that has been reached so far is only $N^{0.280}$ for 1000 bits N (the upper bound on d less for higher number of bits). The basic idea relied on LLL algorithm, but the experimental bounds were constrained by large lattice dimensions. In this paper we present theoretical results as well as experimental evidences to extend the bound of d for which RSA is weak. This requires the knowledge of a few most significant bits of p (alternatively these bits need to be searched exhaustively). We provide experimental results to highlight that the problem can be solved with low lattice dimensions in practice. Our strategy outperforms the existing experimental results by increasing the bounds of d. We provide clear evidence that RSA, with d of the order of $N^{0.3}$ for 1000 bit N, can be cryptanalysed in practice from the knowledge of N, e.

Keywords: Cryptanalysis, Factorization, Lattice, LLL Algorithm, RSA, Weak Keys.

1 Introduction

RSA [12] is one of the most popular cryptosystems in the history of cryptology. Here, we use the standard notations in RSA as follows: primes p, q, with $q < p < 2q$; $N = pq$, $\phi(N) = (p-1)(q-1)$; e, d are such that $ed = 1 + k\phi(N)$, $k \geq 1$; N, e are available in public and the message M is encrypted as $C = M^e \bmod N$; the secret key d is required to decrypt the message as $M = C^d \bmod N$.

Wiener [16] showed that if one uses $d < \frac{1}{3}N^{0.25}$, then RSA is insecure. Boneh and Durfee [3] extended this bound up to $d < N^{0.292}$ using Coppersmith's technique [6]. There exist considerable amount of references in the literature where the bound on d is increased till $O(N^{0.5})$ depending on different constraints on the differences of primes or the values of d, e (see [11, 2, 17] and the references therein). However, there is no general result and experimental evidence where the bound for d can be increased exceeding $O(N^{\delta})$, where for example $\delta = 0.3$. Here we present ideas to achieve better bounds on δ such that N can be factorized from the knowledge of e when d is $O(N^{\delta})$. This is to note that in [13], it has been clearly pointed out that Wiener's method cannot be extended with good efficiency beyond d of the order of $N^{0.25}$. In [8], RSA cryptanalysis has been studied following the idea of [6], where it was considered that some bits of d are known.

In this paper we concentrate on the existing techniques [3, 4, 1] with the idea that a few MSBs of the prime p is known. We consider that some estimate p_0 of p is known such that

$|p - p_0| < N^\gamma$, $\gamma \leq \frac{1}{2}$. That is $(\frac{1}{2} - \gamma) \log_2 N$ many MSBs of p are known. The other way of interpreting it is that one may need to try for $N^{\frac{1}{2}-\gamma}$ many possible options to guess the MSBs of p. With this idea, we find that it is possible to exceed the bound of d over the works of Boneh-Durfee [3, 4] and Blömer-May [1] with low lattice dimensions as used in [3, 4, 1]. Our theoretical results can be summarized as follows. Let $N = pq$, where p and q are primes of same bitsize. Let $d = N^\delta$. Suppose, $p_0 \geq \sqrt{N}$ be an approximation of p with $|p - p_0| < N^\gamma$, $\gamma \leq \frac{1}{2}$. We show that, RSA is insecure if

- $\delta < \frac{\gamma+3-2\sqrt{\gamma(\gamma+3)}}{3}$ (Theorem 3),
- $\delta < 1 - \sqrt{\gamma}$ (Theorem 4), and
- $\delta < \frac{\sqrt{16\gamma^2-4\gamma+4}-(6\gamma-2)}{5}$ (Theorem 5),

modifying the ideas of [3, 4, 1] respectively.

The idea of [3] uses the full rank lattice for attacking this problem. Later in [4], sub-lattices have been used for better results. This idea has been further extended in [1]. The main idea used in [3, 4, 1] and in this work relies on three important parts: (i) reduction of lattice or sub-lattice, (ii) calculation of resultant, (iii) finding roots of the resultant polynomial. The idea of using sub-lattices (with lesser lattice dimension than the full rank lattice) instead of full rank lattice provides improvements in time complexity during the first step, i.e., if sub-lattice is used instead of lattice the requirement of time is less. This we detail in experimental results. However, we have observed that the calculation of resultant needs significantly more time than the first step irrespective of using lattice or sub-lattice. This shows that though the idea of sub-lattices [4, 1] improved the bound on d than in [3] theoretically, there is not much improvement in experimental results due to the overhead in calculating the resultant. This has been pointed out in [1, Section 6] too.

The outline of the paper is as follows. In Section 1.1, we briefly discuss some background materials. Next, in Section 2 we present our strategy on a theoretical framework which is in the line of [3, 4, 1]. Section 3 describes the complete experimental details with comparison of existing works.

1.1 Preliminaries

Now we briefly present some basics on basis reduction in lattice (see [3, 6] and the references therein for more details). Consider that $u_1, \ldots, u_w \in Z^n$ are linearly independent vectors with $w \leq n$. A Lattice, spanned by $< u_1, \ldots, u_w >$, is the set of all linear combinations of $u_1, \ldots, u_w$, i.e., w is the dimension of the lattice. A lattice is called full rank when $w = n$. Let L be a lattice spanned by linearly independent vectors $u_1, \ldots, u_w$, where $u_1, \ldots, u_w \in Z^n$. By $u_1^*, \ldots, u_w^*$, we denote the vectors obtained by applying the Gram-Schmidt process to the vectors $u_1, \ldots, u_w$. It is known that given a basis $u_1, \ldots, u_w$ of a lattice L, LLL algorithm can find a new basis $b_1, \ldots, b_w$ of L with the following properties.

- $\| b_i^* \|^2 \leq 2 \| b_{i+1}^* \|^2$, for $1 \leq i < w$
- For all i, if $b_i = b_i^* + \sum_{j=1}^{i-1} \mu_{i,j} b_j^*$ then $|\mu_{i,j}| \leq \frac{1}{2}$ for all j.

– $\| b_1 \| \leq 2^{\frac{w}{2}} \det(L)^{\frac{1}{w}}$, $\| b_2 \| \leq 2^{\frac{w}{2}} \det(L)^{\frac{1}{w-1}}$.

The determinant of L is defined as $det(L) = \prod_{i=1}^{w} ||u_i^*||$, where $||.||$ denotes the Euclidean norm on vectors.

Let us now explain the issue of solving the small inverse problem as presented in [3]. Let $d < N^{\delta}$. We assume e is same order of magnitude as N. As e gets reduced, the Boneh-Durfee technique [3] works better. Thus for the worst case scenario, one can assume $d < e^{\delta}$. It has been noticed that $ed = 1 \bmod \frac{\phi(N)}{2}$. So $ed + k(\frac{N+1}{2} + s) = 1$, where $k \in Z$, $s = -\frac{p+q}{2}$, i.e., $k(\frac{N+1}{2} + s) - 1 = 0 \bmod e$. Let $f(x, y) = x(N + 1 - y) - 1$. We have to find x_0, y_0 such that $f(x_0, y_0) \equiv 0 (\mathrm{mod} e)$, where, $|x_0| < e^{\delta}$ and $|y_0| < e^{0.5}$. To find the roots, the modular equation is transformed to an equation over integers by the idea of Coppersmith [6]. Given a polynomial $g(x, y) = \sum a_{i,j} x^i y^j$, we define the norm as $\| g(x, y) \|^2 = \sum a_{i,j}^2$.

Theorem 1. *[9] Let $g(x, y)$ be a polynomial which is a sum of ω many monomials. Suppose $g(x_0, y_0) = 0 \bmod e^m$ for some positive integer m, where $|x_0| < X$ and $|y_0| < Y$. If $\| g(xX, yY) \| < \frac{e^m}{\sqrt{\omega}}$, then $g(x_0, y_0) = 0$ holds over integers.*

Following [3], one can define the polynomials $g_{i,k}(x, y) = x^i f^k(x, y) e^{m-k}$ and $h_{j,k}(x, y) = y^j f^k(x, y) e^{m-k}$ for a given positive integer m and $k = 0, \ldots, m$, $i = 0, \ldots, m - k$ and $j = 0, \ldots, t$ for some positive integer t. Now consider the lattice L_B spanned by the coefficient vectors of the polynomials $g_{i,k}(xX, yY)$ and $h_{j,k}(xX, yY)$. One can check that the basis vectors $B(m, t)$ of the lattice L_B form a triangular matrix M_B.

Now we present the definition of geometrically progressive matrices following [4].

Definition 1. *Let M be an $(a+1)b \times (a+1)b$ matrix. The pair (i, j) corresponds to $(bi+j)$-th column of M. Similarly a pair (k, l) can be used to index $(bk + l)$-th row of M.*

Let $C, D, c_0, c_1, c_2, c_3, c_4, \beta$ be real numbers with $C, D, \beta \geq 1$. A matrix M is said to be geometrically progressive with parameters $(C, D, c_0, c_1, c_2, c_3, c_4, \beta)$ if the following conditions hold for all i, k in $[0, \ldots, a]$ and for all j, l in $[1, \ldots, b]$:

– $|M(i, j, k, l)| \leq C \cdot D^{c_0 + c_1 i + c_2 j + c_3 k + c_4 l}$,
– $M(k, l, k, l) = D^{c_0 + c_1 k + c_2 l + c_3 k + c_4 l}$,
– $M(i, j, k, l) = 0$ *whenever* $i > k$ *or* $j > l$.
– $\beta c_1 + c_3 \geq 0$ *and* $\beta c_2 + c_4 \geq 0$.

Theorem 2. *[4, Theorem 5.1] Let M be an $(a + 1)b \times (a + 1)b$ geometrically progressive matrix with parameters $(C, D, c_0, c_1, c_2, c_3, c_4, \beta)$, and let B be a real number. Define*

$$S_B = \{(k, l) \in \{0, \ldots a\} \times \{1, \ldots b\} | M(k, l, k, l) \leq B\}$$

and set $w = |S_B|$. If L is the lattice defined by rows $(k, l) \in S_B$ of M, then

$$det(L) \leq ((a + 1)b)^{w/2} (1 + C)^{w^2} \prod_{(k,l) \in S_B} M(k, l, k, l).$$

Blömer-May [1] referred to the coefficient vectors of the polynomials $g_{i,k}(xX, yY)$ as the X block. The X block is further divided into $(m+1)$ many blocks named as X_l for $l = 0, \ldots, m$, where the block X_l consists of the $l+1$ many coefficient vectors of $g_{i,k}$ with $i+k = l$. Fixing l, each of these $l+1$ vectors is denoted as $X_{l,k}$, $0 \leq k \leq l$ (the k-th vector in the X_l block). That is, $X_{l,k}$ is the coefficient vector of $g_{l-k,k}$.

Further, a Y_j block is defined as the block of all $m+1$ coefficient vectors of the polynomials that are shifted by y^j. The k-th vector in Y_j block is called $Y_{j,k}$, which is the coefficient vector of $h_{j,k}$.

All column vectors with label $x^l y^j$, $l \geq j$ form a column block named $X^{(l)}$. Similarly the column block $Y^{(l)}$ contains all column vectors labeled with $x^i y^{i+l}$. Then a new lattice L_M is presented in [1] as follows.

- Lattice parameters m and t are chosen to build a lattice basis $B(m, t)$.
- In the block Y_t of $B(m, t)$, every vector is removed except for the last vector $Y_{t,m}$. In the block $Y_{(t-1)}$ of $B(m, t)$, every vector is removed except for the last two vectors $Y_{t-1,m}, Y_{t-1,m-1}$. This continues upto the block Y_1, where every vector is removed except the last t vectors $Y_{1,m}, Y_{1,m-1}, \ldots, Y_{1,m-t+1}$.
- Every vector in the block X is removed except for the vectors in the $t+1$ many blocks $X_{m-t}, \ldots, X_m$.
- The columns need to be deleted in such a manner that the resulting basis is again triangular. All column blocks $X^{(0)}, X^{(1)}, \ldots, X^{(m-t-1)}$ are removed. Moreover, in the column block $Y^{(l)}$, $1 \leq l \leq t$, the columns labeled with $x^i y^{i+1}$, $0 \leq i < m - t + l$, are removed.

Below, we give an example of the lattice basis for the parameter choice $m = 3, t = 1$. For this m, t, the basis vectors $B(3, 1)$ of the lattice L_B form a triangular matrix M_B as follows. In case of [3], the product of the diagonal elements gives $det(M_B)$.

	$\downarrow$	$\downarrow$	$\downarrow$											$\downarrow$	$\downarrow$	$\downarrow$	
	1	x	xy	x^2	x^2y	x^2y^2	x^3	x^3y	x^3y^2	x^3y^3	y	xy^2	x^2y^3	x^3y^4
$\rightarrow e^3$	e^3													
$\rightarrow xe^3$		e^3X												
$\rightarrow fe^2$	–	–	e^2XY											
x^2e^3				e^3X^2										
xfe^2		–		–	e^2X^2Y									
f^2e	–	–	–	–	–	eX^2Y^2								
x^3e^3							e^3X^3							
x^2fe^2				–			–	e^2x^3Y						
xf^2e		–		–	–		–	–	eX^3Y^2					
f^3	–	–	–	–	–	–	–	–	–	X^3Y^3				
$\Rightarrow ye^3$											e^3Y			
$\Rightarrow yfe^2$			–								–	e^2XY^2		
$\Rightarrow yf^2e$			–		–	–					–	–	eX^2Y^3	
yf^3			–			–		–	–	–	–	–	–	X^3Y^4

In case of [4], the rows marked by $\Rightarrow$ are removed. In that case, the sub-matrix of M_B is not a square matrix and the determinant is calculated following [4, Theorem 5.1] (presented above in Theorem 2 also). The work of [1] removes the rows marked by $\Rightarrow$ a well as $\rightarrow$. This

sub-matrix of M_B is again not a square one, but the columns marked by $\downarrow$ are also removed to get a square matrix (see [1, Theorems 2, 3] for more details).

It has been demonstrated in [3] that for $\delta < 0.284$, one can find m, t such that N can be factored using the LLL algorithm. Further the idea was improved to extend this bound upto 0.292 by using non-triangular lattice bases [4]. Improvements towards implementation have been studied in [1] by significantly reducing the lattice dimension for same m, t. They could achieve the bound of δ till $N^{0.290}$ theoretically which is less than the Boneh-Durfee bound of $N^{0.292}$, but the results of [1] were more efficient in practice.

Our idea in this paper is to extend the bounds of [3, 4, 1] further with small lattice dimension given a few MSBs of p (which can also be searched exhaustively). One may note that given the constraint $q < p < 2q$, a few MSBs of p can be known efficiently (see Section 3.3). This will indeed reduce the search effort further.

It is time consuming to handle large lattice dimensions and that is the constraint in extending the value of δ using Boneh-Durfee [3] and related techniques [4, 1]. Also it is not very clear how large lattice dimensions can be handled efficiently in a parallel environment. In our case, it is very easy to distribute the work in different machines independently for different choices of the MSBs, given the small lattice dimension to work with. Advantage of our ideas in comparison with [3, 4, 1] is presented in Section 3.

2 Achieving higher upper bounds on d

We start working in the direction of [3, Section 4].

Theorem 3. *Let $N = pq$, where p and q are primes of same bitsize. Let $d = N^\delta$. Suppose, $p_0 \geq \sqrt{N}$ be an approximation of p with $|p - p_0| < N^\gamma$, $\gamma \leq \frac{1}{2}$. We show that, RSA is insecure if $\delta < \frac{\gamma+3-2\sqrt{\gamma(\gamma+3)}}{3}$.*

Proof. We assume $e = N$ as for $e < N$ one can get better upper bound on δ (similar to the approach of [3, Page 9]).

Let $q_0 = \frac{N}{p_0}$. We have $ed = 1+k\phi(N) = 1+k(N+1-p-q) = 1+k(N+1-p_0-q_0-(p+q-p_0-q_0)) = 1+x(A+y)$, where $x = k < d = N^\delta = e^\delta, A = N+1-p_0-q_0, y = -(p+q-p_0-q_0)$. As $p > \sqrt{N}$ and as we assume $p_0 \geq \sqrt{N}$ too, we have $|y| < N^\gamma = e^\gamma$.

We have to find x_0, y_0 such that $1 + x_0(A + y_0) \equiv 0 \bmod e$, where $|x_0| < e^\delta$ and $|y_0| < e^\gamma$. Let $X = e^\delta, Y = e^\gamma$. Note that we consider the same X as in [3, Section 4], but our Y is generalized as Y has been taken as $e^{\frac{1}{2}}$ in [3, Section 4].

One may refer to [3, Section 4] for $det_x = e^{m(m+1)(m+2)/3} X^{m(m+1)(m+2)/3} Y^{m(m+1)(m+2)/6}$ and $det_y = e^{tm(m+1)/2} X^{tm(m+1)/2} Y^{t(m+1)(m+t+1)/2}$. Plugging in the values of X and Y (note that our Y is different than [3, Section 4]), we obtain, $det_x = e^{m^3(\frac{1}{3}+\frac{\delta}{3}+\frac{\gamma}{6})+o(m^3)}$, $det_y = e^{tm^2(\frac{1}{2}+\frac{\delta}{2}+\frac{\gamma}{2})+t^2m\frac{\gamma}{2}+o(tm^2)}$. Now $det(L) = det_x det_y$ and we need to satisfy $det(L) < e^{mw}$, where $w = (m + 1)(m + 2)/2 + t(m + 1)$, the dimension of L. To satisfy $det(L) < e^{mw}$, we need $m^3(\frac{1}{3}+\frac{\delta}{3}+\frac{\gamma}{6})+tm^2(\frac{1}{2}+\frac{\delta}{2}+\frac{\gamma}{2})+t^2m\frac{\gamma}{2} < (tm+\frac{m^2}{2})m$. This leads to $m^2(-\frac{1}{6}+\frac{\delta}{3}+\frac{\gamma}{6})+tm(-\frac{1}{2}+\frac{\delta}{2}+\frac{\gamma}{2})+t^2\frac{\gamma}{2} < 0$. After fixing an m, the left hand side is minimized at $t = \frac{\frac{1}{2}-\frac{\delta}{2}-\frac{\gamma}{2}}{\gamma}m$. Putting

this value we have, $m^2(-\frac{1}{6}+\frac{\delta}{3}+\frac{\gamma}{6})+\frac{m^2(-\frac{1}{2}+\frac{\delta}{2}+\frac{\gamma}{2})(\frac{1}{2}-\frac{\delta}{2}-\frac{\gamma}{2})}{\gamma}+\frac{(\frac{1}{2}-\frac{\delta}{2}-\frac{\gamma}{2})^2m^2}{\gamma^2}\frac{\gamma}{2}<0$, simplying $(-\frac{1}{6}+\frac{\delta}{3}+\frac{\gamma}{6})+\frac{(-\frac{1}{2}+\frac{\delta}{2}+\frac{\gamma}{2})(\frac{1}{2}-\frac{\delta}{2}-\frac{\gamma}{2})}{\gamma}+\frac{(\frac{1}{2}-\frac{\delta}{2}-\frac{\gamma}{2})^2}{\gamma^2}\frac{\gamma}{2}<0$. Hence, $\delta<\frac{2\gamma+6-\sqrt{(2\gamma+6)^2+12(\gamma^2+2\gamma-3)}}{6}$ and simplifying we get $\delta<\frac{\gamma+3-2\sqrt{\gamma(\gamma+3)}}{3}$.

Similar to the idea presented in [3, Section 4], if the first two elements (polynomials $P_1(x,y)$, $P_2(x,y)$) of the reduced basis out of the LLL algorithm are algebraically independent (i.e., nonzero resultant $res(P_1,P_2)$ which is a polynomial of y, say), then we will get x_0, y_0 correctly which will in turn provide the factorization of N making RSA insecure. (This actually happens with a high probability in practice as we have also checked by experimentation.) □

First note that the result presented in Theorem 3 gives the same upper bound 0.284 presented in [3] when $\gamma=\frac{1}{2}$. In the above theorem we use lattice of full rank. We can improve the bounds on δ if we use the techniques based on sub-lattice [4, 1]. These ideas are discussed in Section 2.1.

Based on Theorem 3, one can design

- a probabilistic polynomial time algorithm $\mathcal{A}$, which will take
- N, e, p_0 as inputs
- and will provide correct p if
  - $|p-p_0|<N^\gamma$,
  - $\delta<\frac{\gamma+3-2\sqrt{\gamma(\gamma+3)}}{3}$,
  - and the resultant polynomial $res(P_1,P_2)$ on y is nonzero with integer solution (in practice the integer solution is correct with a high probability).

It is important to study the performance of the algorithm $\mathcal{A}$, based on different values of m, t. Our main improvement is achieved due to the lesser bound on Y. Once again, we like to reiterate that we consider the same X as in [3, Section 4], but our $Y=e^\gamma$ is smaller than the Y considered as $e^{\frac{1}{2}}$ in [3, Section 4].

The knowledge of MSBs of p can be interpreted as following also. One can use $\mathcal{A}$ for $p_0=\sqrt{N}$ to $\sqrt{2N}$ (as for $q<p<2q$, we have $\sqrt{N}<p<\sqrt{2N}$) at an interval of N^γ, i.e., one needs to try for $(\sqrt{2}-1)N^{\frac{1}{2}-\gamma}$ many steps. Each step will require $\pi(N)$ time complexity when $\pi(N)$ is the running time for $\mathcal{A}$. It is important to mention here that proper choice of m, t are required to get the results and we will compare this with [3, Section 6, Page 9] and [1, Table 1, Section 6, Page 18].

For $\gamma=0.477$, we find that $\delta<0.30044$, and for $\gamma=0.478$, we find that $\delta<0.29975$. Thus, for $\delta=0.3$, it is enough to consider $\gamma=0.477$, which gives $\frac{1}{2}-\gamma=0.023$. Hence, theoretically speaking, we either need to know $(\frac{1}{2}-\gamma)\times\log_2 N$ many bits of p_0 or $N^{\frac{1}{2}-\gamma}$ many invocations of $\mathcal{A}$ are required. Thus, N can be factorized in $O(N^{0.023}\pi(N))$ time complexity with the knowledge of e when d is $O(N^{0.3})$. For 1000 bit integers, our strategy will require either the knowledge of 23 bits or 2^{23} invocations of $\mathcal{A}$. See Table 1 later in Section 2.1 for more detailed results. However, these are only theoretical estimates and we will present the exact experimental details in Section 3.

Remark 1. If we do not neglect the lower order terms in the proof of Theorem 3, then to satisfy $det(L) < e^{mw}$ we need

$$m(m+1)(\frac{m+2}{3}+\frac{t}{2})\delta+(m+1)(\frac{m(m+2)}{6}+\frac{t(m+t+1)}{2})\gamma < m(m+1)(\frac{m+2}{6}+\frac{t}{2}). \quad (1)$$

The value of γ is always $\frac{1}{2}$ in the analysis of [3]. However, due to the knowledge of a few bits, we can have $\gamma < \frac{1}{2}$, and thus it is possible to get extended bound on δ in our case for the same m, t. This is the reason we get improved bounds on d from Theorem 3 than the idea of [3], for same lattice dimension. Later, in Section 2.1, we exploit the ideas related to sub-lattice. In such cases also, without neglecting the lower order terms in the proofs of Theorems 4, 5, one can get better bounds on d than what presented in [4, 1] respectively. Experimental results supporting this idea is presented in Section 3.

2.1 Exploiting the Sub-lattice based Techniques

Boneh and Durfee showed how they improve their result $\delta < 0.284$ [3] to $\delta < 0.292$ [4] using the sub-lattice technique. We will now follow the idea of [4]. This idea has also been followed in [17, Section 6].

Theorem 4. *Let $N = pq$, where p and q are primes of same bitsize. Let $d = N^{\delta}$. Suppose, $p_0 \geq \sqrt{N}$ be an approximation of p with $|p-p_0| < N^{\gamma}$, $\gamma \leq \frac{1}{2}$. We show that, RSA is insecure if $1-2\gamma < \delta < 1-\sqrt{\gamma}$.*

Proof. This proof is similar to the proof of Theorem 3, till the calculation of det_x. However, det_y will be different here than in the proof of Theorem 3.

Let M_{By} be the portion of the matrix M_B with rows corresponding to the y shifts and columns corresponding to the variables of the form $x^u y^v$, for $v > u$. In this case, M_{By} is a geometrically progressive matrix with parameter choice $(m^{2m}, e, m, \delta+\gamma, \gamma-1, -1, 1, b)$ for some b. One may note that the first three conditions of Definition 1 hold. To satisfy the fourth condition, the parameter b should satisfy $b(\delta+\gamma)-1 \geq 0$ and $b(\gamma-1)+1 \geq 0$ together and thus we get the constraint $\delta > 1-2\gamma$, which in turn gives a possible value of b as $b = \frac{2}{2-2\gamma}$. Similar to the idea of [4], we also get the optimal choice for t as twice the value of t in Theorem 3, i.e., $t = \frac{1-\delta-\gamma}{\gamma}m$. Following Definition 1, we have $M_{By}(k,l,k,l) = e^{m+(\delta+\gamma-1)k+\gamma l}$. Denote S_B (as in Theorem 2) by S when $B = e^m$. By our choice of t, we have $(k,l) \in S$ iff $l \leq \frac{1-\delta-\gamma}{\gamma}k$. Neglecting the lower order terms, $|S| = \frac{1-\delta-\gamma}{2\gamma}m^2$. Thus $w = \frac{(m+1)(m+2)}{2} + |S| = \frac{m^2}{2} + |S| = (\frac{1}{2} + \frac{1-\delta-\gamma}{2\gamma})m^2$ (neglecting lower order terms) $= \frac{1-\delta}{2\gamma}m^2$. Following the similar idea as in [4] and going through similar calculation in [17, Section 6] for the sub-lattice, we get $det_y = e^{\frac{1}{12}\frac{9-4(\delta+\frac{1}{2}+\gamma)^2}{2\gamma}m^3}$. Then the condition $det(L) = det_x det_y < e^{mw}$ gives the bound $\delta < 1-\sqrt{\gamma}$. □

The result presented in Theorem 4 gives the same upper bound 0.292 presented in [4] when $\gamma = \frac{1}{2}$.

Next we present an approach following [1, Section 4].

Theorem 5. *Let $N = pq$, where p and q are primes of same bitsize. Let $d = N^{\delta}$. Suppose, $p_0 \geq \sqrt{N}$ be an approximation of p with $|p - p_0| < N^{\gamma}$, $\gamma \leq \frac{1}{2}$. We show that, RSA is insecure if $\delta < \frac{\sqrt{16\gamma^2 - 4\gamma + 4} - (6\gamma - 2)}{5}$.*

Proof. This proof is again similar to the proof of Theorem 3, but both det_x and det_y will be different here than in the proof of Theorem 3. Given that certain rows and columns of M_B will be removed following the idea of [1], the diagonal elements of the new matrix will be

$X^m e^m, X^m Y e^{m-1}, \ldots, X^m Y^m$,
$X^{m-1} e^m, X^{m-1} Y e^{m-1}, \ldots, X^{m-1} Y^{m-1} e$,
$\ldots$,
$X^{m-t} e^m, X^{m-t} Y e^{m-1}, \ldots, X^{m-t} Y^{m-t} e^t$
for x-shifts (i.e., they will contribute to det_x) and
$X^m Y^{m+t}$,
$X^m Y^{m+t-1}, X^{m-1} Y^{m+t-2} e$,
$\ldots$,
$X^m Y^{m+1}, X^{m-1} Y^m e, \ldots X^{m-t+1} Y^{m-t+2} e^{t-1}$,
for y-shifts (i.e., they will contribute to det_y).

Multiplying the diagonal elements and neglecting the lower order terms, we need the condition

$$X^{tm^2 - \frac{mt^2}{2} + \frac{t^3}{6}} Y^{\frac{tm^2}{2} + \frac{t^3}{6}} < e^{\frac{tm^2}{2}}.$$

Putting the values of $X = e^{\delta}, Y = e^{\gamma}, t = \tau m$, we have the required condition

$$(\frac{\delta}{6} + \frac{\gamma}{6})\tau^2 - \frac{1}{2}\delta\tau + (\delta + \frac{\gamma}{2} - \frac{1}{2}) < 0.$$

The left hand side is minimum when $\tau = \frac{\delta}{\frac{2}{3}(\delta + \gamma)}$. Putting this value of τ, in the previous inequality we get the bound on δ. □

The result presented in Theorem 5 gives the same upper bound 0.290 presented in [1] when $\gamma = \frac{1}{2}$.

In Table 1 we present the corresponding values of γ for which the values of δ can be reached. The value of $\frac{1}{2} - \gamma$ gives the proportion of bits we need to know or search exhaustively. We start listing the results from $\delta = 0.285$, as already there is theoretical result available for $\delta = 0.284$ using full rank lattice [3] (the theoretical result achieving $\delta = 0.292$ is presented using sub-lattice in [4]).

δ	γ Theorem 3	γ Theorem 4	γ Theorem 5
0.285	0.49962	0.5	0.5
0.290	0.49224	0.5	0.49985
0.295	0.48491	0.49703	0.49284
0.300	0.47763	0.48999	0.48589

Table 1. Theoretical estimates of γ following Theorems 3, 4, 5 to reach the corresponding bounds on δ.

It is clear from the Table 1, that from theoretical point of view, the best efficiency is achieved in Theorem 4, followed by Theorem 5 and Theorem 3.

3 Complete Experimental Details

We have implemented the program in SAGE 2.10.1 over Linux Ubuntu 7.04 on a computer with Dual CORE Intel(R) Pentium(R) D CPU 2.80GHz, 1 GB RAM and 2 MB Cache. While comparing our results to the existing results [3, 4, 1], we will present higher bounds on d indeed.

In Section 3.1, we present the results related to the algorithms in [3, 4, 1] on our platform. Then we discuss the implementation results in Section 3.2 related to Theorems 3, 4, 5.

3.1 Existing experimental results

First we restate the results of [4, 1] in Table 2 (left) to give an idea of lattice dimensions required for certain δ values. The time estimates are presented as in the corresponding papers [4, 1].

N	δ	m	t	lattice dimension	running time	Reference
1000 bits	0.270	6	2	21	19 minutes	[1]
1000 bits	0.274	8	3	36	300 minutes	[1]
1000 bits	0.2765	10	4	55	26 hours	[1]
1000 bits	0.278	11	5	72	6 days	[1]
1000 bits	0.280	7	3	45	14 hours	[4]
2000 bits	0.265	4	2	15	6 minutes	[1]
2000 bits	0.275	7	3	45	65 hours	[4]
4000 bits	0.265	5	2	25	14 hours	[4]
6000 bits	0.265	4	2	15	100 minutes	[1]
6000 bits	0.269	5	2	18	8 hours	[1]
10000 bits	0.255	3	1	11	90 minutes	[4]

N	m	t	w	Reference	T_l	T_r	T_s
1000 bits	7	3	60	[3]	76.29	411.42	3.90
1000 bits	7	3	45	[4]	61.17	410.72	4.20
1000 bits	7	3	32	[1]	29.58	333.76	2.46
2000 bits	7	3	60	[3]	347.56	1139.58	11.03
2000 bits	7	3	45	[4]	283.69	1147.34	13.17
2000 bits	7	3	32	[1]	148.71	938.79	8.02
4000 bits	7	3	60	[3]	1358.30	2461.80	37.89
4000 bits	7	3	45	[4]	1041.47	2465.66	58.23
4000 bits	7	3	32	[1]	605.03	2028.11	32.96
6000 bits	5	3	39	[3]	129.82	213.68	7.81
6000 bits	5	3	25	[4]	83.27	211.49	6.70
6000 bits	5	3	24	[1]	78.13	218.98	6.88
10000 bits	3	1	14	[3]	1.95	5.35	0.83
10000 bits	3	1	11	[4]	1.60	5.34	0.82
10000 bits	3	1	8	[1]	1.53	5.38	0.82

Table 2. Left: Results from [4, 1]. Right: Experimental Results for execution of the algorithms presented in [3, 4, 1] on our platform for $\delta = 0.26$.

We have already pointed out in the introduction (Section 1) that for practical experiments, the resultant calculation takes more time than lattice reduction. It is not clear from the experimental results in [3, 4] whether the resultant calculation time has been considered. In [1, Section 6], it has been clearly commented that only lattice reduction time has been presented.

Let us denote the lattice reduction time by T_i^l, the resultant calculation time by T_i^r and the solution time by T_i^s (in seconds) for the i-th run.

We have implemented the algorithms of [3, 4, 1] to study all the time requirements in detail. Below we present the running time for 1000, 2000, 4000, 6000 and 10000 bits N and for $\delta = 0.26$. The experiments are executed for a single run in each case. It is clear from the experimental results in Table 2 (right) that the lattice reduction takes minimum time

for [1] and the works of [3, 4] take little more time, but the resultant calculation takes more time than lattice reduction in all the cases and thus the improvements in lattice reduction techniques do not have significant effects for experimental purposes.

Note that, given m, t, the lattice dimension w can be calculated directly as $w = (m + 1)(m + 2)/2 + t(m + 1)$ for [3] (Theorem 3 in our approach) and $w = (m + 1)(t + 1)$ for [1] (Theorem 5 in our approach). However, there is no exact formula for calculating w from m, t for the case of [4] (Theorem 4 in our approach). Thus, in this section, the value of w is presented as found experimentally following Theorem 4 in our approach and we present the maximum value of w, when more than one runs are executed.

3.2 Experimental results based on our approach

In this section we concentrate on the experimental results following our Theorems 3, 4, 5. In Table 3, we present our results for 1000-bit N that shows that the bound on δ can be increased much further than the work of [3, 4, 1] as listed in Table 2 (left).

$m = 3, t = 1, w = 14$ (Theorem 3)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	30	32	31.2	0.60	0.072	0.167	0.016
0.285	38	40	39.2	0.60	0.077	0.242	0.038
0.290	46	48	47.0	0.89	0.077	0.241	0.039
0.295	53	56	53.9	1.04	0.080	0.240	0.038
0.300	59	63	61.3	1.11	0.081	0.242	0.038

$m = 3, t = 1, w = 12$ (Theorem 4)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	31	34	32.1	1.14	0.056	0.164	0.016
0.285	38	40	39.1	0.83	0.057	0.242	0.038
0.290	45	47	46.3	0.64	0.056	0.231	0.037
0.295	53	55	54.1	0.70	0.059	0.244	0.041
0.300	60	62	61.4	0.80	0.061	0.242	0.038

$m = 3, t = 1, w = 8$ (Theorem 5)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	27	31	29.9	0.86	0.037	0.165	0.016
0.285	36	40	38.0	0.84	0.038	0.166	0.016
0.290	43	48	45.8	0.94	0.039	0.166	0.016
0.295	52	56	54.1	0.84	0.040	0.166	0.015
0.300	59	64	62.1	0.99	0.041	0.166	0.015

$m = 5, t = 2, w = 33$ (Theorem 3)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	15	17	16.6	0.48	2.17	21.50	0.57
0.285	23	25	24.2	0.74	2.39	21.59	0.57
0.290	32	33	32.2	0.39	2.51	22.32	0.61
0.295	37	39	38.2	0.74	2.68	22.33	0.56
0.300	45	47	46.6	1.01	2.84	22.45	0.67

$m = 5, t = 2, w = 27$ (Theorem 4)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	16	17	16.4	0.48	1.42	21.34	0.65
0.285	23	25	24.4	1.02	1.45	21.67	0.54
0.290	30	32	31.6	0.79	1.84	22.45	0.51
0.295	39	40	39.2	0.39	2.02	22.48	0.66
0.300	45	48	46.6	1.01	2.14	22.42	0.50

$m = 5, t = 2, w = 18$ (Theorem 5)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	16	18	17.0	0.89	1.22	16.97	0.33
0.285	24	26	25.4	0.80	1.28	16.94	0.28
0.290	31	33	32.2	0.74	1.31	16.76	0.32
0.295	39	41	40.0	0.63	1.74	16.81	0.29
0.300	47	49	48.0	0.48	1.48	16.86	0.28

$m = 5, t = 3, w = 39$ (Theorem 3)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	16	18	16.9	0.83	2.59	21.54	0.58
0.285	23	26	24.8	0.87	2.93	22.08	0.56
0.290	31	33	31.9	0.70	3.06	22.36	0.58
0.295	38	40	38.8	0.60	3.28	22.46	0.56
0.300	45	47	45.6	0.66	3.41	22.42	0.59

$m = 5, t = 3, w = 27$ (Theorem 4)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	16	18	16.7	0.64	1.42	21.59	0.59
0.285	23	26	24.7	0.90	1.76	21.95	0.60
0.290	30	33	31.5	0.81	1.89	22.26	0.62
0.295	38	40	38.6	0.66	2.00	22.44	0.60
0.300	45	48	46.5	0.67	2.11	22.45	0.57

$m = 5, t = 3, w = 24$ (Theorem 5)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	15	17	15.9	0.70	1.56	21.50	0.59
0.285	23	24	23.4	0.49	1.66	21.62	0.57
0.290	30	32	30.7	0.64	1.81	22.40	0.62
0.295	36	38	37.8	0.60	1.93	22.36	0.59
0.300	44	46	44.8	0.75	2.03	22.40	0.57

$m = 7, t = 3, w = 60$ (Theorem 3)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	8	10	8.9	0.70	34.33	449.42	5.74
0.285	16	19	17.0	0.89	36.99	449.31	5.66
0.290	23	26	24.1	0.83	39.02	450.50	5.96
0.295	31	33	31.4	0.66	46.070	456.99	5.87
0.300	37	41	38.7	1.19	47.91	475.73	5.81

$m = 7, t = 3, w = 48$ (Theorem 4)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	8	10	9.0	0.89	26.22	450.46	6.12
0.285	15	18	16.4	0.80	28.48	449.73	5.59
0.290	23	24	23.8	0.40	32.23	449.79	5.55
0.295	30	33	31.8	0.97	36.93	465.58	5.96
0.300	38	40	38.6	0.66	42.44	476.71	6.09

$m = 7, t = 3, w = 32$ (Theorem 5)

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.280	9	11	10.2	0.60	14.91	334.51	2.37
0.285	16	19	17.5	0.90	16.04	335.38	2.41
0.290	25	26	25.4	0.49	16.68	333.98	2.44
0.295	32	34	32.9	0.54	16.93	333.98	2.41
0.300	39	41	40.3	0.64	18.10	334.23	2.33

Table 3. Comparison of Experimental results following Theorems 3, 4, 5

While considering the primes, we take $q < p < 2q$, i.e., p, q are of same bit size. Further we select p, q randomly with the constraint that $p - q > n^{0.45}$ and $2q - p > n^{0.45}$ so that the ideas in the direction of [11, 17] do not work efficiently. The cryptanalytic strategy of [17] works well when $p - q$ is bounded and in the same direction, the method of [11] works well when $2q - p$ is bounded.

By τ-bit N we mean that p, q are of $\frac{\tau}{2}$ bits each. The column with δ provides that we consider the first d which is greater than or equal to $\lceil N^{\delta} \rceil$ such that d is coprime to $\phi(N)$. After fixing δ, the size of N (which is 1000 bits for the Table 3) and the lattice parameters (m, t, w), we go for 100 runs for each case (except for $m = 7, t = 3$, when we go for 10 runs). For each run, we calculate the minimum number of bits (for i-th run, call this variable v_i) that need to be known to factorize N from the knowledge of N, e using the ideas of Theorems 3, 4, 5. We present the minimum $\min(v_i)$, maximum $\max(v_i)$, average $\overline{v}$ and standard deviation $\sigma(v)$ of the data set v_i, for $i = 1, \ldots, 100$, (for $m = 7, t = 3$, when we go for 10 runs, $i = 1, \ldots, 10$). Further, the time requirement is also presented in detail. The data in Table 3 clearly presents the improvements over the works of [3, 4, 1] for achieving higher bounds on δ for 1000 bits N.

We also present the results using the lattice parameters $m = 11, t = 5$. Using the technique of Theorem 3, we get the lattice dimension $w = 138$ and exploiting the strategy using sub-lattice following Theorem 4, we get $w = 105$ at maximum. Among the techniques we discuss in this paper, the minimum sub-lattice dimension is $w = 72$, using the idea of Theorem 5. We present the implementation results for this in Table 4. Due to longer time requirement, we present the result of one run in each case.

$m = 11, t = 5, w = 72$ (Theorem 5)				
δ	v_i	T_l	T_r	T_s
0.280	3	1185.81	16741.77	46.92
0.285	10	1425.12	16954.72	39.11
0.290	18	1630.63	17107.79	40.23
0.295	25	1687.21	17135.02	59.61
0.300	33	1770.66	17222.95	59.38

Table 4. Experimental results following Theorem 5 for 1000-bit N.

A few results are provided for 2000, 4000 and 6000 bits N in Table 5. We consider the δ values which are higher than the results achieved in [4, 1] (see also Table 2, left). To have a comparison, we take same or lower values of m, t as used in the highest values for δ in [3, 4, 1]. For each δ value, we take an average of 10 runs.

Next we present our experimental results for 10000 bits N in Table 6. For each δ value, we take an average of 3 runs.

In [8], RSA cryptanalysis has been studied following the idea of [7], where it was considered that some bits of d are known. The result of [8], that can be compared with this effort is for small values of d. We show that the requirement of knowledge of bits in d as in [8, MSB1 attack, Section 5] is much higher than our requirement of the knowledge of bits in p. Referring [8, Section 5], one may note that for MSB1 attack, 30% of bits need to be known for d of the order of $N^{0.3}$, when N is 1024 bits. Note that d is of 308 bits and 30% of the

N	δ	m	t	w	Theorem No.	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
2000 bits	0.280	7	3	60	3	16	18	17.1	116.24	1233.02	18.44
2000 bits	0.280	7	3	45	4	15	18	16.9	103.36	1235.72	19.81
2000 bits	0.280	7	3	32	5	18	20	19.1	59.88	933.90	8.39
4000 bits	0.270	5	2	33	3	0	1	0.2	16.66	166.75	6.45
4000 bits	0.270	5	2	25	4	0	1	0.3	13.32	166.89	6.35
4000 bits	0.270	5	2	18	5	2	4	3	12.45	132.68	3.39
6000 bits	0.270	5	2	33	3	0	0	0	34.94	279.05	13.57
6000 bits	0.270	5	2	25	4	0	1	0.2	28.90	268.91	14.16
6000 bits	0.270	5	2	18	5	3	4	3.5	26.11	212.93	7.49

Table 5. Our Results for 2000, 4000 and 6000 bits N.

Following Theorem 3, $m = 3, t = 1, w = 14$

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$	δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.260	0	0	0	0	1.95	5.37	.82	0.275	213	215	214.33	0.94	2.32	5.35	.80
0.261	0	0	0	0	1.61	5.32	.82	0.280	293	295	294.0	0.81	2.48	5.40	1.75
0.262	3	4	3.33	0.47	1.64	5.44	.84	0.285	369	371	370.0	0.82	2.62	5.39	1.92
0.263	19	21	20.0	0.82	1.70	5.43	.84	0.290	442	446	443.66	1.70	2.69	7.92	1.92
0.265	52	54	53.0	0.82	1.74	5.35	.81	0.295	517	519	518.0	0.81	2.74	7.94	1.88
0.270	133	134	133.33	0.47	2.16	5.40	.82	0.300	590	590	590.0	0.0	2.76	7.96	1.92

Following Theorem 4, $m = 3, t = 1, w = 12$

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$	δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.260	0	0	0	0	1.60	5.34	.82	0.275	213	216	215.0	1.41	2.22	5.44	.82
0.261	0	0	0	0	1.40	5.36	.82	0.280	294	295	294.66	0.47	2.38	7.86	1.88
0.262	3	5	4.0	0.82	1.42	5.29	.84	0.285	369	370	369.33	0.45	2.42	7.93	1.94
0.263	20	21	20.33	0.47	1.44	5.37	.81	0.290	443	444	443.6	0.48	2.47	7.90	1.85
0.265	52	53	52.66	0.47	1.49	5.38	.80	0.295	517	519	518.0	0.81	2.52	7.92	1.87
0.270	133	134	133.33	0.47	2.07	5.40	.84	0.300	590	592	591.0	0.81	2.72	7.89	1.98

Following Theorem 5, $m = 3, t = 1, w = 8$

δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$	δ	$\min(v_i)$	$\max(v_i)$	$\overline{v}$	$\sigma(v)$	$\overline{T}_l$	$\overline{T}_r$	$\overline{T}_s$
0.260	0	0	0	0	1.53	5.35	.83	0.275	211	214	213.0	1.41	1.48	5.48	.82
0.261	0	0	0	0	1.36	5.38	.82	0.280	291	294	292.33	1.25	1.49	5.39	.82
0.262	1	3	2	0.82	1.32	5.35	.84	0.285	373	375	374.33	0.94	1.56	5.40	.82
0.263	17	20	18.33	1.24	1.35	5.40	.83	0.290	453	455	454.0	0.82	1.58	5.39	.80
0.265	50	52	51.00	0.82	1.36	5.41	.82	0.295	534	536	535.0	0.81	1.64	5.43	.81
0.270	131	132	131.33	0.47	1.42	5.36	.82	0.300	616	617	616.33	0.49	1.69	5.38	.81

Table 6. Our results for 10000 bits N.

bits of d is 94 bits. The result of [8] involved a reduction of 30-dimensional lattice. Instead of going for the knowledge of some bits in d, we consider the knowledge of a few bits of p. For 1024 bits N and 308 bits d, considering lattice dimensions 39 ($m = 5, t = 3$) and 27 ($m = 5, t = 1$) which need around 27, 23 seconds to resolve respectively, we require only around 50, 57 bits (much less than 94 bits required in [8] for d) of p to be known respectively.

3.3 Time requirement for the complete attack

When the information about a few of the MSBs of the prime p is not known, exhaustive search is the only option to try for those bits. Thus it is advantageous for the complete attack if one can find out a few MSBs of p by some basic techniques.

In [15], some heuristics have been proposed to estimate a few MSBs of p from N, by studying the Continued Fraction (CF) expression of $\frac{1}{\sqrt{N}}$. Empirical results have been presented to show that for 1024-bit N, one can find out around 7 MSBs of p. However, the method requires to test approximately 25 many values for good probability of success. One may refer to [15, Page 122], where the experimental results suggest a standard deviation of 12 and hence approximately $12 + 1 + 12 = 25$ values need to be searched around the mean value. As, $2^4 < 25 < 2^5$, the actual advantage is approximately $7 - 5 = 2$ bits.

As the MSB of p is 1, the search effort is always reduced by 1 bit, when the bit size of p is known. Since $q < p < 2q$, we have $2\sqrt{N} < p + q < \frac{3}{\sqrt{2}}\sqrt{N}$. One simple approach is to divide the range $(2\sqrt{N}, \frac{3}{\sqrt{2}}\sqrt{N})$ in 2^r many values to get an estimate of $p + q$. Then solving $N = pq$, and the estimate of $p + q$, the MSBs of p can be estimated. Clearly, the estimation of MSBs will be better in case of $p + q$ than that of p.

For experimental evidence, we considered 1000 bits N and tried to estimate a few MSBs of p. We divided the range $(2\sqrt{N}, \frac{3}{\sqrt{2}}\sqrt{N})$ in 2^5 many values and then considering the best of all the 32 options for the MSBs of p, we find that around 7.75 many MSBs can be known considering the average of 10,000 runs. This gives an advantage of $7.75 - 5 = 2.75$ bits in search. Further, 8 to 14 MSBs of p are available in the proportions 0.54, 0.27, 0.14, 0.07, 0.04, 0.02, 0.01 respectively. This gives a reduction of $14 - 5 = 9$ bits in search, when the probability of success is 0.01, i.e., 1%.

Based on these observations, we estimate the time required for actual attack on our platform. Note that we present results with the δ values greater than those reported in [3, 4, 1]. The best experimental δ values of [3, 4, 1] are presented in Table 2 (left). Thus, the δ values, we present here, have never been reported in published materials.

- Let us first consider $\delta = 0.285$ for 1000 bit N. Using $m = 7, t = 3, w = 48$, we find from Table 3 that around 16 MSBs of p need to be known to cryptanalyze RSA. Thus, we need 2^{15} many attempts (the MSB is always known). Given each run requires around 484 seconds following the idea of Theorem 4, we need only a week with a cluster of 26 machines. Given that 9 MSBs of p will be known with success probability 1%, this requires around 17 hours on a single machine.
- Now we consider $\delta = 0.3$ for 1000 bit N. Using $m = 7, t = 3, w = 48$, we find from Table 3 that around 38 MSBs of p need to be known to cryptanalyze RSA. Thus, we need 2^{37} many attempts (the MSB is always known). Given each run requires around 524 seconds following the idea of Theorem 4, we need around 398 days with a cluster of 2^{21} machines. Given that 9 MSBs of p will be known with success probability 1%, this requires a cluster of 2^{13} machines for little more than a year.
- Now we consider $\delta = 0.3$ for $m = 11, t = 5$ following Table 4. Here the lattice dimension is 72 only. Our results in Table 4 identifies that we need around 6 hours following the idea of Theorem 5 to get the result in a machine we have referred (i.e. around 4 runs in a day). The requirements of bits will be around 33 for $\delta = 0.3$. The search effort will be 2^{32} as the MSB of p is always 1. Thus we need a cluster of 2^{21} many CPUs to complete the task in 512 days. Given that 9 MSBs of p will be known with success probability 1%, this requires 256 days with 2^{13} machines.
- For 2000 bits N, we present experimental results when $\delta = 0.280$. We need to search 17 bits on an average. Following Table 5, we find that around 64 many runs can be completed in a day using the ides of Theorem 4. Thus, the total work can be completed in a day with 2^{10} computers with our specifications.
- We could reach the bound $\delta = 0.270$ for 4000 and 6000 bits N, with the knowledge of very few bits (in some cases without the knowledge of any bit) in p.

– For 10000 bits N, we could reach the bound $\delta = 0.261$ without the knowledge of any bit in p. Moreover, from the results in Table 6, it is clear that the bound of $\delta = 0.263$ can be achieved in practice by searching around 20 MSBs of p.

A closer look at Tables 3, 4, 5, 6 points out that the idea of Theorem 4 provides the most efficient results during implementation among Theorems 3, 4, 5, i.e., the technique presented in [4] works most efficiently among the ideas of [3, 4, 1] when taken into our paradigm of guessing a few MSBs of p.

4 Conclusion

In this paper we show that the techniques of [3, 4, 1] can be modified to have higher bounds on d with low lattice dimensions. First of all, our idea provides theoretical extension of the bound of $N^{0.292}$ given the knowledge of some MSBs of p, which can also be managed by exhaustive search. We use the same lattice dimensions as presented in [3, 4, 1] to have larger values of d for which RSA can be attacked given N, e. Our experimental results outperform the state of the art results of [3, 4, 1] for 1000, 2000, 4000, 6000 and 10000 bits N. We justify that for 1000 bits N, RSA can be crypanalyzed in practice when d is of the order of $N^{0.3}$.

References

1. J. Blömer and A. May. Low secret exponent RSA revisited. CaLC 2001, LNCS 2146, pp. 4–19, 2001.
2. J. Blömer and A. May. A generalized Wiener attack on RSA. PKC 2004, LNCS 2947, pp. 1–13, 2004.
3. D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. Eurocrypt 1999, LNCS 1592, pp. 1–11, 1999.
4. D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. IEEE Trans. on Information Theory, 46(4):1339–1349, 2000.
5. H. Cohen. A Course in Computational Algebraic Number Theory. Springer Verlag, 1996.
6. D. Coppersmith. Small solutions to polynomial equations and low exponent vulnerabilities. Journal of Cryptology, 10(4):223–260, 1997.
7. J. S. Coron. Finding Small Roots of Bivariate Integer Equations Revisited. Proc. of Eurocrypt 2004, LNCS 3027, pp. 492–505.
8. M. Ernst, E. Jochemsz, A. May and B. de Weger. Partial key exposure attacks on RSA up to full size exponents. Eurocrypt 2005, LNCS 3494, pp. 371–386, 2005.
9. N. Howgrave-Graham. Finding small roots of univariate modular equations revisited. Proceedings of Cryptography and Coding, LNCS 1355, pp. 131-142, 1997.
10. A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen, vol. 261, pp. 515-534, 1982.
11. S. Maitra and S. Sarkar. Revisiting Wiener's Attack – New Weak Keys in RSA. 11th International Conference, ISC 2008, LNCS 5222, 2008.
12. R. L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. Communications of ACM, 21(2):158–164, Feb. 1978.
13. R. Steinfeld, S. Contini, J. Pieprzyk and H. Wang. Converse results to the Wiener attack on RSA. PKC 2005, LNCS 3386, pp. 184–198, 2005.
14. D. R. Stinson. Cryptography – Theory and Practice. 2nd Edition, Chapman & Hall/CRC, 2002.
15. H. -M. Sun, M. -E. Wu and Y. -H. Chen. Estimating the prime-factors of an RSA modulus and an extension of the Wiener attack. ACNS 2007, LNCS 4521, pp. 116–128, 2007.
16. M. Wiener. Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory, 36(3):553–558, 1990.
17. B. de Weger. Cryptanalysis of RSA with small prime difference. Applicable Algebra in Engineering, Communication and Computing, 13(1):17–28, 2002.