CRYPTANALYSIS OF RSA WITH LATTICE ATTACKS

BY

ANDREW HOON SUK

B.S., University of Illinois at Urbana-Champaign, 2001

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois
ABSTRACT

In this paper we will analyze several attacks on the RSA cryptosystem. We start with a brief introduction of the highly used cryptosystem and its fundamentals. Then we will go into lattice theory and the LLL algorithm. Next we describe the Boneh-Durfee attack on RSA with short secret keys. We will show ways to improve upon their attack and propose a new attack with a better bound for short vectors. Finally, we will discuss other possible applications such as factorization.
# Table of Contents

Chapter 1  INTRODUCTION ................................................. 1
  1.1 Overview of This Paper ........................................... 2
  1.2 Notation .............................................................. 3
    1.2.1 Norms of polynomials ....................................... 3

Chapter 2  THE RSA CRYPTOSYSTEM ................................... 5
  2.1 Symmetric and Asymmetric Cryptosystems ......................... 5
  2.2 How RSA Works ...................................................... 6
  2.3 Recalling Old Theorems ........................................... 7
    2.3.1 The extended Euclidean algorithm ......................... 8
  2.4 RSA Key Generation ................................................ 9
  2.5 Previous Attacks on Short Secret Key ........................... 9
    2.5.1 Wiener’s attack ............................................. 9
    2.5.2 Boneh-Durfee attack ...................................... 10

Chapter 3  LATTICE THEORY .............................................. 11
  3.1 Lattices .............................................................. 11
  3.2 The LLL Algorithm .................................................. 11
    3.2.1 The LLL algorithm ......................................... 12
  3.3 Properties of LLL ................................................... 12
  3.4 Finding Small Modular Solutions ................................ 14

Chapter 4  THE BONEH DURFEE ATTACK ............................... 16
  4.1 Small Inverse Problem ............................................ 16
  4.2 Building a Lattice .................................................. 17
  4.3 Geometrically Progressive Matrices .............................. 19
  4.4 Bounding the Determinant of the New Lattice ................. 21
  4.5 Boneh-Durfee Revisited .......................................... 23

Chapter 5  HIGH ORDER BITS .............................................. 24
  5.1 Boneh-Durfee Attack with High Bits Known ................. 24
  5.2 Geometrically Progressive Matrices ............................ 27
  5.3 Bounding the Determinant of the New Lattice ................ 28
  5.4 Methods on Obtaining High Order Bits ....................... 29
  5.5 Experiments ......................................................... 30
  5.6 Conclusion .......................................................... 32
Chapter 1

INTRODUCTION

Cryptology is the scientific study of cryptography and cryptanalysis. Cryptography is the enciphering and deciphering of messages in secret code or cipher. The idea is to easily encrypt a message such that it is very hard to decrypt without a secret key. Cryptanalysis is the solving of cryptograms or cryptographic systems. The art of Cryptology has been studied for thousands of years, and has drastically changed over the last hundred.

In the early 1900s the German’s introduced the Enigma Machine, which played a crucial role in the Second World War. The Enigma Machine permuted the letters of the English alphabet with three scrambler disks and a plugboard. The three scrambler disks had 26 possible orientations, with six possible ways to order them. Then the plugboard swapped any pair of letters giving the enigma machine roughly $10^{16}$ possible keys. The cipher appeared to be unbreakable. However at the Government Code and Cypher School in Bletchley Park, the Enigma Machine was finally broken by a team of code breakers lead by Alan Turing. The Enigma Machine had a weakness that was one of the main problems in cryptography, key exchange. The question was, Could two parties exchange a secret key over a public channel without exposing it?

In 1976, Martin Hellman and Whitfield Diffie of Stanford University solved the problem by introducing the Diffie-Hellman key exchange in their seminal paper New Directions in Cryptography. The idea uses one-way functions, which are calculations easy to perform but really hard to undo. A typical one-way function is

$$M^x \mod(P),$$

where $M, P, x \in \mathbb{Z}$. $M^x \mod(P)$ is a one-way function because it is easy to compute, but given
Alice and Bob agree on a public key \( M, P \in \mathbb{Z} \), i.e., \( M = 7, P = 11 \).

Bob chooses a secret integer \( B \), i.e., \( B = 6 \).

Bob computes the one-way function \( \beta = M^B \mod(P) \), i.e., \( \beta = 7^6 \mod(11) = 4 \).

Bob sends \( \beta \) over publicly.

Step 3

Alice sends \( \alpha \) over publicly.

Step 4

Alice takes \( \beta \) and computes \( k = \alpha^\beta \mod(P) \), i.e., \( k = 4^6 \mod(11) = 64 \mod(11) = 9 \).

$k = 2^6 \mod(11) = 64 \mod(11) = 9$

Euclidean Division Algorithm

\[
\begin{align*}
\alpha &= 7^3 \mod(11) = 2 \\
\beta &= 7^6 \mod(11) = 4 \\
\end{align*}
\]

1.1 Overview of This Paper

In this paper, we discuss how we can apply lattice theory to attack the RSA cryptosystem.
1.2 Notation

Table 1.1 shows a summary of several symbols used throughout the paper. The vector spaces $\mathbb{Z}^n$, $\mathbb{Q}^n$ and $\mathbb{R}^n$ uses the standard inner product: For $\{b_1, ..., b_n\}$ as the standard basis\(^1\) for $\mathbb{Z}^n$, let

$$ (u, v) = \sum_{i=1}^{n} \xi_i \eta_i, $$

where $u = \sum \xi_i b_i$, $v = \sum \eta_i b_i$ and $\xi, \eta \in \mathbb{Z}$. Likewise for $\mathbb{Q}^n$ and $\mathbb{R}^n$. Hence we use the common Euclidean norm:

$$ ||u||^2 = \sum_{i=1}^{n} \xi_i^2. $$

Table 1.1: Special notations used throughout the paper.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{Z}$</td>
<td>Ring of integers</td>
</tr>
<tr>
<td>$\mathbb{Q}$</td>
<td>Field of rational numbers</td>
</tr>
<tr>
<td>$\mathbb{R}$</td>
<td>Field of real numbers</td>
</tr>
<tr>
<td>$\mathbb{Z}^n$</td>
<td>Vector space of $n$-tuples over the integers</td>
</tr>
<tr>
<td>$\mathbb{Q}^n$</td>
<td>Vector space of $n$-tuples over the rational numbers</td>
</tr>
<tr>
<td>$\mathbb{R}^n$</td>
<td>Vector space of $n$-tuples over the real numbers</td>
</tr>
<tr>
<td>$\mathbb{Z}[x]$</td>
<td>Ring of univariate polynomials over the integers</td>
</tr>
<tr>
<td>$\mathbb{Z}[x, y]$</td>
<td>Ring of bivariate polynomials over the integers</td>
</tr>
<tr>
<td>$\mathbb{Z}[x, y, z]$</td>
<td>Ring of three variable polynomials over the integers</td>
</tr>
</tbody>
</table>

1.2.1 Norms of polynomials

Given a polynomial $g(x) = \sum_i a_i x^i$, we define the norm of $g(x)$ as

$$ ||g(x)||^2 = \sum_i a_i^2. $$

For a positive real number $X \in \mathbb{R}$, we define the weighted norm as

$$ ||g(xX)||^2 = \sum_i (a_i X^i)^2. $$

\(^1\)For nonnegative integers $i \leq n$, $b_i = (0, ..., 0, 1, 0, ..., 0)$ where the 1 is in the $i$th position.
For the bivariate case, given \( h(x, y) = \sum_{i,j} b_{i,j} x^i y^j \), we define the norm to be

\[
||h(x, y)||^2 = \sum_{i,j} b_{i,j}^2.
\]

For positive real numbers \( X, Y \in \mathbb{R} \), we define the weighted norm as

\[
||h(xX, yY)||^2 = \sum_{i,j} (b_{i,j} X^i Y^j)^2.
\]

For the 3 variable case, given \( p(x, y, z) = \sum_{i,j,k} c_{i,j,k} x^i y^j z^k \), we define the norm to be

\[
||p(x, y, z)||^2 = \sum_{i,j,k} c_{i,j,k}^2.
\]

For positive real numbers \( X, Y, Z \in \mathbb{R} \), we define the weighted norm as

\[
||p(xX, yY, zZ)||^2 = \sum_{i,j,k} (c_{i,j,k} X^i Y^j Z^k)^2.
\]

Finally we always refer to \( e \in \mathbb{Z} \) as the public exponent in the RSA cryptosystem, never the constant \( \approx 2.78 \).
Chapter 2

THE RSA CRYPTOSYSTEM

RSA was the first public key cryptosystem and was introduced in 1977. In this chapter, we will give a brief description of the cryptosystem and it’s fundamentals.

2.1 Symmetric and Asymmetric Cryptosystems

Definition 2.1 An encryption scheme or cryptosystem is a tuple \((\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})\) with the following properties:

1: \(\mathcal{P}\) is a set called the plaintext space. All elements in \(\mathcal{P}\) are plaintexts.

2: \(\mathcal{C}\) is a set called the ciphertext space. All elements in \(\mathcal{C}\) are called ciphertexts.

3: \(\mathcal{K}\) is the set called the key space. All elements in \(\mathcal{K}\) are called keys.

4: \(\mathcal{E} = \{E_k : k \in \mathcal{K}\}\) is the family of functions \(E_k : \mathcal{P} \mapsto \mathcal{C}\). All elements in \(\mathcal{E}\) are called encryption functions.

5: \(\mathcal{D} = \{D_k : k \in \mathcal{K}\}\) is the family of functions \(D_k : \mathcal{C} \mapsto \mathcal{P}\). All elements in \(\mathcal{D}\) are called decryption functions.

6: For each \(e \in \mathcal{K}\), there exists a \(d \in \mathcal{K}\) such that \(D_d(E_e(p)) = p\) for all \(p \in \mathcal{P}\).

Cryptosystems with encryption key \(e\) that is equal to decryption key \(d\) is called symmetric. Cryptosystems where keys \(d\) and \(e\) are different are called asymmetric, and the computation of \(d\) from \(e\) is relatively hard. The RSA cryptosystem is based on an asymmetric cryptosystem, making the encryption key \(e\) public and decryption key \(d\) private.
2.2 How RSA Works

Here is a simple step-by-step mathematical explanation of the RSA public key cryptosystem:

Step 1: Alice selects two large prime integers $p$ and $q$.

Step 2: Alice multiplies them together such that $N = pq$. Then she picks her second public key $e \in \mathbb{Z}$, such that $e$ and $(p - 1)(q - 1)$ are relatively prime.

Step 3: Alice now publishes $(N, e)$. This is known as Alice’s public key. If anyone wants to send Alice a secret message, they use her public key to encrypt the message.

Step 4: The message must be converted into an integer $M \in \mathbb{Z}$. This can be done by changing the text to ASCII\(^1\) binary digits. $M$ is then encrypted into a ciphertext $C \in \mathbb{Z}$ by

$$C = M^e \pmod{N}.$$  

Step 5: So, if Bob wants to send Alice the letter X, then X is converted to ASCII as $M = 1011000$.

Step 6: To encrypt this message, Bob looks up Alice’s public key and obtains $(N, e)$. Hence then Bob computes

$$C = M^e \pmod{N}.$$  

Notice exponentials in modular arithmetic are one-way functions, so it is very difficult to work backwards from $C$. Now when $C$ is sent out publicly to Alice, Eve can not decipher the message.

Step 7: Now Alice can decipher the message with her secret key $d \in \mathbb{Z}$. The term $d$ is calculated by

$$ed = 1 \pmod{(p-1)(q-1)}.$$  

---

\(^1\)American Standard Code for Information Interchange.
Hence the \( \gcd(e, (p - 1)(q - 1)) = 1 \), and such a number \( d \) exists. It can be computed by the extended Euclidean algorithm.

**Step 8:** For Alice to decrypt the message, she simply calculates

\[
M = C^d \pmod{N}.
\]

As you can see, the RSA cryptosystem relies heavily on the discrete log and factoring problem.

### 2.3 Recalling Old Theorems

Here we will show such an integer \( d \) exists, and how to compute it.

**Fact 2.1** If \( \gcd(a, m) = 1 \), then for every integer \( b \), the congruence

\[
ax \equiv b \pmod{m}
\]

can be solved for \( x \).

**Proof:** Since \( \gcd(a, m) = 1 \), there exists an integer \( s \) such that \( asb \equiv b \pmod{m} \), so that \( x = sb \) is a solution. \( \blacksquare \)

Remember \( \gcd(e, (p - 1)(q - 1)) = 1 \). So for \( ed \equiv 1 \pmod{(p - 1)(q - 1)} \), the fact applies and we can solve for \( d \). In order to compute \( d \), we must introduce the following:

**Theorem 2.1** (extended Euclidean algorithm). Let \( a \) and \( b \) be positive integers. There is an algorithm that finds \( d = \gcd(a, b) \), and a pair of integers \( s \) and \( t \) such that \( d = sa + tb \).

**Proof:** Note we can take \( a \) and \( b \) to be positive since the general cases follows

\[
\gcd(-a, b) = \gcd(-a, -b) = \gcd(a, -b) = \gcd(a, b).
\]

The basic idea of the algorithm is to keep repeating the division algorithm. Let \( b = r_0 \) and \( a = r_1 \). Then the repeated sequence follows:
\[ b = q_1 r_1 + r_2 \quad r_2 < r_1 = a \]
\[ r_1 = q_2 r_2 + r_3 \quad r_3 < r_2 \]
\[ r_2 = q_3 r_3 + r_4 \quad r_4 < r_3 \]
\[ \ldots \]
\[ r_{n-3} = q_{n-2} r_{n-2} + r_{n-1} \quad r_{n-1} < r_{n-2} \]
\[ r_{n-2} = q_{n-1} r_{n-1} + r_n \quad r_n < r_{n-1} \]
\[ r_{n-1} = q_n r_n \]

where we have \( q_j \) and \( r_j \) known from the division algorithm. Now we have \( d = r_n \) since for every common divisor \( c \), \( c \mid d \). By going backwards, we have

\[ d = r_n = r_{n-2} - q_{n-1} r_{n-1} \]

which is a linear combination of \( r_{n-2} \) and \( r_{n-1} \). Combining this with the equation above we have a linear combination of \( r_{n-2} \) and \( r_{n-3} \). Keep repeating this procedure leads us to \( d = sa + tb \). ■

The extended Euclidean algorithm has two non-negative integers \( a \) and \( b \) as inputs, and outputs \( d = \gcd(a,b) \) and integer \( x, y \) such that \( d = ax + by \). Below is a pseudo code for the algorithm.

### 2.3.1 The extended Euclidean algorithm

1: If \( b = 0 \) then \( d \leftarrow a, x \leftarrow 1 \) and \( y \leftarrow 0 \).

2: Set \( x_2 \leftarrow 1, x_1 \leftarrow 0, y_2 \leftarrow 0 \) and \( y_1 \leftarrow 1 \).

3: While \( b > 0 \), do:
   
   1: \( q = \lfloor a/b \rfloor, r \leftarrow a - qb, x \leftarrow x_2 - qx_1 \) and \( y \leftarrow y_2 - qy_1 \).
   
   2: \( a \leftarrow b, b \leftarrow r, x_2 \leftarrow x_1, x_1 \leftarrow x, y_2 \leftarrow y_1 \) and \( y_1 \leftarrow y \).
   
   3: end loop.

4: Set \( d \leftarrow a, x \leftarrow x_2, y \leftarrow y_2 \) and return \((d, x, y)\).
The algorithm has a running time of $O((\log n)^2)$ where $a, b \leq n$. Notice with Theorem 2.1, one can break RSA by the following fact:

**Fact 2.2** If one can efficiently factor the modulus $N = pq$, then one can efficiently recover $d$.

**Proof:** If one knows $p$ and $q$, then $(p - 1)(q - 1)$ can easily be computed. Since $e$ is public, with the Extended Euclidean Algorithm, one can compute $d$. ■

### 2.4 RSA Key Generation

For an $n$-bit encryption, the key generation algorithm finds two random $\frac{n}{2}$-bit prime integers such that

$$\frac{\sqrt{N}}{2} < q < p < 2\sqrt{N}.$$ 

We call $p$ and $q$ balanced primes, since they have the same order. The key generation algorithm then selects integers $e$ and $d$ such that $ed \equiv 1 \mod(\phi(N))$. Although in the previous section Alice chose $e$ first, she could just as well have selected $d$ first. Sometimes $e$ is selected to a desired integer (i.e., $e = 3$), and then $d$ is computed with the extended Euclidean algorithm. However many times $d$ is selected first, in order to make it small, and then $e$ is computed with the extended Euclidean algorithm. Then the public exponent $e$ is usually the same size as $N$, roughly $n$-bits.

### 2.5 Previous Attacks on Short Secret Key

Here we summarize two previous attacks on RSA with balanced primes $p$ and $q$, and short secret key $d$.

#### 2.5.1 Wiener’s attack

Wiener’s attack [2] is based on the continued fractions algorithm. For two integers $A$ and $B$ that satisfy $|x - \frac{B}{A}| < \frac{1}{2A^2}$ where $x$ is a known rational, $\frac{B}{A}$ can be recovered in polynomial time. So Wiener looked at the equation $ed \equiv 1 \mod((p - 1)(q - 1)/2)$ which is equivalent to

$$ed = 1 + k\left(\frac{N + 1}{2} - s\right), \quad (2.1)$$
where $k, d \in \mathbb{Z}$ and $s = (p + q)/2$ are unknown. Hence we have

$$|\frac{2e}{N} - \frac{k}{d}| = \frac{|2 + k(1 - 2s)|}{Nd},$$

and we need to satisfy

$$\frac{k(2s - 1) - 2}{N} < \frac{1}{2d}.$$

Notice when $e$ and $N$ have the same order, $k$ and $d$ have the same order as well. For some secret key $d$, we set $k \approx d = N^\delta$ and $s \approx N^{0.5}$. Ignoring small order terms, we have

$$ksd \approx N^\delta N^{0.5} N^\delta = N^{2\delta + 0.5} < N.$$

Hence for $\delta < 0.25$, RSA is insecure.

2.5.2 Boneh-Durfee attack

Dan Boneh and Glenn Durfee of Stanford University were able to improve Wiener’s old bound with a heuristic attack for secret key $d < N^{0.292}$ [3]. This paper is a direct extension of the Boneh-Durfee attack, and will be analyzed in Chapter 4.
Chapter 3

LATTICE THEORY

In this chapter we will give a brief introduction on lattices and it’s properties. Most importantly we will discuss the LLL algorithm. Named after the discoverers Lenstra, Lenstra, and Lovasz [4], the LLL algorithm produces a lattice basis of relatively short vectors in polynomial time. The algorithm has been heavily used for designing attacks on various cryptosystems.

3.1 Lattices

Let \( b_1, \ldots, b_m \in \mathbb{Z}^n \) be linearly independent vectors with integers \( m \leq n \). A lattice \( L \) spanned by \( \langle b_1, \ldots, b_m \rangle \) is the set of all integer combinations of \( b_1, \ldots, b_m \). The lattice is at full rank if \( m = n \). We denote \( b_1^*, \ldots, b_m^* \) as the vectors obtained by applying the Gram-Schmidt orthogonalization process to the vectors \( b_1, \ldots, b_m \). We define the determinant of the lattice \( L \)

\[
\det(L) = \prod_{i=1}^{m} \|b_i^*\|.
\]

Now with the help of linear algebra, a lattice is at full rank implies

\[
\det(L) = \det(M),
\]

where \( M \) is the \( n \times n \) matrix whose rows are the vectors \( b_1, \ldots, b_m \).

3.2 The LLL Algorithm

The LLL algorithm inputs linearly independent row vectors \( B = (b_1, \ldots, b_n) \subseteq \mathbb{Z}^n \), and outputs a reduced basis \( (r_1, \ldots, r_n) \subseteq L = \sum_i \mathbb{Z}b_i \subseteq \mathbb{Z}^n \). We label \( R^* \) the Gram-Schmidt orthogonalization
(GSO) of $R$, and $T$ is an $n \times n$ matrix over the rational such that $TR = R^*$. For $1 \leq j < i$, we label $\mu_{ij}$ as

$$\mu_{ij} = \frac{r_i \ast r_j^*}{\|r_j^*\|^2},$$

where $\ast$ is the usual inner product of two vectors in $\mathbb{Z}^n, \mathbb{Q}^n$, or $\mathbb{R}^n$. We write for $\mu \in \mathbb{R}$, $[\mu] = [\mu + 1/2]$ as the nearest integer. Below is a pseudo code for the algorithm.

### 3.2.1 The LLL algorithm

1: \textbf{for} $i = 1, \ldots, n$ \textbf{do} $r_i \leftarrow b_i$

\hspace{1cm} compute the (GSO) $R^*, M \in \mathbb{Q}^{n \times n}$. $i \leftarrow 2$.

2: \textbf{while} $i \leq n$ \textbf{do}

\hspace{1cm} 1: \textbf{for} $j = i - 1, i - 2, \ldots, 1$, \textbf{do} $r_i \leftarrow r_i - \lceil \mu_{ij} \rceil r_j$, update and overwrite the old GSO.

\hspace{1cm} 2: \textbf{if} $i > 1$ and $\|r_{i-1}^*\|^2 > 2\|r_i^*\|^2$

\hspace{1.5cm} then exchange $r_{i-1}$ and $r_i$ and update and overwrite the GSO, $i \leftarrow i - 1$

\hspace{1cm} \textbf{else} $i \leftarrow i + 1$.

3: \textbf{return} $r_1, \ldots, r_n$.

For a lattice of rank $= w$, the LLL algorithm uses $O(w^4 \log A)$ arithmetic operations on integers whose length is $O(w \log A)$, where $A = \max \|v_i\|$.

### 3.3 Properties of LLL

Here we will examine some of the properties of the reduced vectors produced by the LLL algorithm.

**Fact 3.1** The LLL algorithm takes $\langle b_1, \ldots, b_w \rangle$ as input, and outputs $\langle r_1, \ldots, r_w \rangle$ such that

1: $\|r_i^*\|^2 \leq 2\|r_{i+1}^*\|^2$ for $1 \leq i < w$. 
2: \( r_i = r_i^* + \sum_{j=1}^{i-1} \mu_j r_j^* \), then \(|\mu_j| < \frac{1}{2} \ \forall i, j\).

Now we create the following bounds.

**Fact 3.2** For a lattice \( L \) spanned by \(<b_1, ..., b_w>\), the LLL algorithm runs in polynomials time and produces a new basis \(<r_1, ..., r_w>\) of \( L \) such that

\[ ||r_1|| \leq 2^{-w/2} \det(L)^{1/w}. \]

**Proof:** We know \( r_1 = r_1^* \), so

\[ \det(L) = \prod_i ||r_i^*|| \geq 2^{-w/2} ||r_1||^w. \]

Jutla [5] then provided a method to bound \( r_2 \). Here we give a sharper bound for \( r_2 \) rather than the one found in [3]. For a given lattice basis \(<b_1, ..., b_w>\), we define

\[ \lambda_{\text{min}}^* = \min_i ||b_i^*||. \]

**Fact 3.3** For reduced vectors \(<r_1, ..., r_w>\),

\[ ||r_2|| \leq \max \left( 2^{-w/2} \det(L)^{1/2}, 2^{w/2} \left[ \frac{\det(L)}{\lambda_{\text{min}}^*} \right]^{1/(w-1)} \right). \]

**Proof:** Here, \( \lambda_{\text{min}}^* \) is the lower bound of the shortest vector in the lattice. Notice

\[ \det(L) = \prod_i ||r_i^*|| > ||r_1^*|| \cdot ||r_2^*||^{w-1} 2^{-2(w-1)/2} \geq \lambda_{\text{min}}^* \cdot ||r_2^*||^{w-1} 2^{-2(w-1)/2}. \]

Hence

\[ ||r_2^*|| < 2^{(w-1)/2} \left[ \frac{\det(L)}{\lambda_{\text{min}}^*} \right]^{1/(w-1)}, \]

which implies

\[ ||r_2||^2 \leq ||b_2^*||^2 + \frac{1}{4} ||r_1||^2 \leq 2^{(w-1)} \left[ \frac{\det(L)}{\lambda_{\text{min}}^*} \right]^{2/(w-1)} + 2^{w-2} \det(L)^{1/2} \]

\[ \leq \max \left( 2^{w-1} \det(L)^{1/2}, 2^{w} \left[ \frac{\det(L)}{\lambda_{\text{min}}^*} \right]^{2/(w-1)} \right). \]

\[ \blacksquare \]
3.4 Finding Small Modular Solutions

Howgrave-Graham [6] showed how to solve small roots of modular equations. For the bivariate case, we have the following lemma.

**Lemma 3.1** Let \( h(x, y) \in \mathbb{Z}[x, y] \) be a polynomial which is a sum of at most \( w \) monomials. Suppose that

a. \( h(x_0, y_0) = 0 \mod e^m \) for some positive integer \( m \) where \( |x_0| < X \) and \( |y_0| < Y \), and

b. \( ||h(xX, yY)|| < e^m / \sqrt{w} \),

then \( h(x_0, y_0) = 0 \) holds over the integers.

**Proof:** Notice that

\[
|h(x_0, y_0)| = \left| \sum a_{i,j} x_0^i y_0^j \right| = \left| \sum a_{i,j} X^i Y^j (\frac{x_0}{X})^i (\frac{y_0}{Y})^j \right| \leq \\
\leq \sum \left| a_{i,j} X^i Y^j (\frac{x_0}{X})^i (\frac{y_0}{Y})^j \right| \leq \sum \left| a_{i,j} X^i Y^j \right| \leq \sqrt{w} ||h(xX, yY)|| < e^m.
\]

Since \( h(x_0, y_0) = 0 \mod e^m, h(x_0, y_0) = 0. \)

For the three variable case we have the following lemma.

**Lemma 3.2** Let \( h(x, y, y) \in \mathbb{Z}[x, y] \) be a polynomial, which is a sum of at most \( w \) monomials. Suppose that

a. \( h(x_0, y_0, z_0) = 0 \mod u^m \) for some positive integer \( m \) where \( |x_0| < X, |y_0| < Y \) and \( |z_0| < Z \)

b. \( ||h(xX, yY, zZ)|| < u^m / \sqrt{w} \)

Then \( h(x_0, y_0, z_0) = 0 \) holds over the integers.

**Proof:** Notice that
\[ |h(x_0, y_0, z_0)| = \left| \sum a_{i,j,k} x_0^i y_0^j z_0^k \right| = \left| \sum a_{i,j,k} x^i y^j z^k \left( \frac{x}{x_0} \right)^i \left( \frac{y}{y_0} \right)^j \left( \frac{z}{z_0} \right)^k \right| \leq \]
\[ \leq \sum \left| a_{i,j,k} x^i y^j z^k \left( \frac{x}{x_0} \right)^i \left( \frac{y}{y_0} \right)^j \left( \frac{z}{z_0} \right)^k \right| \leq \sum |a_{i,j,k} x^i y^j z^k| \leq \leq \sqrt{w} \sum \max |h(x, y, z)| < u^m. \]

Since \( h(x_0, y_0, z_0) = 0 \mod u^m \), \( h(x_0, y_0, z_0) = 0 \).
Chapter 4

THE BONEH DURFEE ATTACK

At Eurocrypt ’99, Dan Boneh and Glenn Durfee [3] presented a lattice attack against RSA for secret key $d < N^{0.292}$. Although the attack is only a heuristic, it performs very well. After nearly 10 years, this was the first significant improvement over Wiener’s bound of secret key $d < N^{0.25}$. This chapter is a complete summary of their paper [3].

4.1 Small Inverse Problem

Remember RSA public key is a pair of integers $\langle N, e \rangle$ where $N = pq$. The corresponding private key is an integer $d$ such that it satisfies $ed = 1 \mod \frac{\phi(N)}{2}$ where $\phi(N) = N - p - q + 1$. For $s = -\frac{(p+q)}{2}$ and $A = \frac{N+1}{2}$, notice

$$k(A + s) \equiv 1 \pmod{e}.$$ 

We write $e = N^\alpha$ and $d = N^\delta$ for some $\alpha$ and $\delta \in \mathbb{R}$. For balanced prime $p$ and $q$, we have $p, q < 2\sqrt{N}$. Notice that we can bound $s$ and $k$ by

$$|k| < \frac{2de}{\phi(N)} \leq 3de/N < 3e^{1+\frac{\delta-1}{\alpha}}$$

and

$$|s| < 2N^{0.5} = 2e^{1/(2\alpha)}.$$ 

When we ignore small order terms and take $\alpha \approx 1$, we have the following problem: find integers $k$ and $s$ such that
\[ k(A + s) \equiv 0 \pmod{e} \]

where
\[ |s| < e^{0.5} \text{ and } |k| < e^\delta. \]

This is known as the \textit{small inverse problem}.

### 4.2 Building a Lattice

The \textit{small inverse problem} now is the following: given a polynomial \( f(x, y) = x(A + y) - 1 \), find \((x_0, y_0)\) such that
\[ f(x_0, y_0) \equiv 0 \pmod{e} \]

where
\[ |x_0| < e^\delta = X \text{ and } |y_0| < e^{0.5} = Y. \]

This implies that we are looking for polynomials with small norm that has \((x_0, y_0)\) as a root modulo \(e^m\). So Boneh and Durfee defined the following:

\[ g_{i,k}(x, y) = x^i f^k(x, y)e^{m-k} \]

and

\[ h_{j,k}(x, y) = y^j f^k(x, y)e^{m-k}. \]

Given an integer \(m\), the lattice \(L\) is spanned by the coefficient vectors of the polynomials for \(k = 0, \ldots, m\). For each \(k\) we use \(g_{i,k}(x_X, y_Y)\) for \(i = 0, \ldots, m - k\) and use \(h_{j,k}(x_X, y_Y)\) for \(j = 0, \ldots, t\) where \(t\) is \textit{minimized} based on \(m\). We refer the vectors generated by the coefficients of \(g_{i,k}(x_X, y_Y)\) as the \textit{x-shifts}, and \(h_{j,k}(x_X, y_Y)\) as the \textit{y-shifts}. This results in a full rank triangular matrix with \(w = (m + 1)(m + 2)/2 + t(m + 1)\) rows. Table 4.1 is the Boneh-Durfee lattice with \(m = 3\) and \(t = 1\). The “-” symbols denote nonzero entries whose values we do not care about. The determinant of the lattice can be easily computed as the product of the diagonal entries.
Table 4.1: The matrix spanned by $g_{i,k}$ and $h_{j,k}$ for $k = 0, 3, i = 0, 3 - k$, and $j = 0, 1$. 

<table>
<thead>
<tr>
<th>$x^3$</th>
<th>$x^3$</th>
<th>$x^3$</th>
<th>$x^3$</th>
<th>$x^3$</th>
<th>$x^3$</th>
<th>$x^3$</th>
<th>$x^3$</th>
<th>$x^3$</th>
<th>$x^3$</th>
<th>$x^3$</th>
<th>$x^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^2$</td>
<td>$x^2$</td>
<td>$x^2$</td>
<td>$x^2$</td>
<td>$x^2$</td>
<td>$x^2$</td>
<td>$x^2$</td>
<td>$x^2$</td>
<td>$x^2$</td>
<td>$x^2$</td>
<td>$x^2$</td>
<td>$x^2$</td>
</tr>
<tr>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
<td>$x$</td>
</tr>
<tr>
<td>$y$</td>
<td>$y$</td>
<td>$y$</td>
<td>$y$</td>
<td>$y$</td>
<td>$y$</td>
<td>$y$</td>
<td>$y$</td>
<td>$y$</td>
<td>$y$</td>
<td>$y$</td>
<td>$y$</td>
</tr>
<tr>
<td>$x^2y$</td>
<td>$x^2y$</td>
<td>$x^2y$</td>
<td>$x^2y$</td>
<td>$x^2y$</td>
<td>$x^2y$</td>
<td>$x^2y$</td>
<td>$x^2y$</td>
<td>$x^2y$</td>
<td>$x^2y$</td>
<td>$x^2y$</td>
<td>$x^2y$</td>
</tr>
<tr>
<td>$xy^2$</td>
<td>$xy^2$</td>
<td>$xy^2$</td>
<td>$xy^2$</td>
<td>$xy^2$</td>
<td>$xy^2$</td>
<td>$xy^2$</td>
<td>$xy^2$</td>
<td>$xy^2$</td>
<td>$xy^2$</td>
<td>$xy^2$</td>
<td>$xy^2$</td>
</tr>
<tr>
<td>$y^3$</td>
<td>$y^3$</td>
<td>$y^3$</td>
<td>$y^3$</td>
<td>$y^3$</td>
<td>$y^3$</td>
<td>$y^3$</td>
<td>$y^3$</td>
<td>$y^3$</td>
<td>$y^3$</td>
<td>$y^3$</td>
<td>$y^3$</td>
</tr>
</tbody>
</table>

\[
\det_x = e^{m(m+1)(m+2)/3} \cdot X^{m(m+1)(m+2)/3} \cdot Y^{m(m+1)(m+2)/6}
\]

and

\[
\det_y = e^{tm(m+1)/2} \cdot X^{tm(m+1)/2} \cdot Y^{r(m+1)(m+t+1)/2},
\]

where $\det_x$ is the determinant due to the $x$-shifts and $\det_y$ is due to the $y$-shifts. When we plug in the values for $X$ and $Y$, we get

\[
\det_x = e^{m(m+1)(m+2)(5+4\delta)/12}
\]

and

\[
\det_y = e^{tm(m+1)(1+\delta)/2+t(m+1)(M+t+1)/4}.
\]

For reduced polynomial $r_1(x, y)$ to satisfy Lemma 3.1, we need to satisfy

\[
\det(L) = \det_x \cdot \det_y < e^{mw}/\gamma,
\]

(4.1)

where $\gamma = (w/2^w)^w/2$. Since $w$ is only a function of $\delta$ and not the public exponent $e$, $\gamma$ is negligible compared to $e^{mw}$. When we ignore small order terms we need to satisfy $\det(L) = \det_x \cdot \det_y < e^{mw}$, which implies

\[
m(m+1)(m+2)^{5+4\delta}/12 + tm(m+1)^{1+\delta}/2 + t(m+1)(m+t+1)/4 < \frac{m(m+1)(m+2)}{2} + tm(m+1),
\]

which leads to

18
\[ m(m + 2)(-1 + 4\delta) + 3tm(-1 + 2\delta) + 3(t + 1) < 0. \]

We obtain the minimum value on the left-hand-side for \( t = \frac{m(1 - 2\delta) - 1}{2} \). Plugging this equation in, we have

\[ -(3 + 2m + 7m^2) + \delta(28m^2 + 20m) - 12m^2\delta^2 < 0, \]

which implies

\[
\delta < \frac{7}{6} - \frac{1}{3}\sqrt{\frac{7}{m} + \frac{4}{m^2} + \frac{5}{6m}}. \tag{4.2}
\]

As \( m \) approaches infinity, this value converges to

\[ \delta < 0.28747. \]

### 4.3 Geometrically Progressive Matrices

The value \( Y \) along the diagonal of the Boneh-Durfee matrix is heavily populated in the \( y \)-shifts portion. By eliminating all the vectors in the \( y \)-shifts that exceeded \( e^m \) along the diagonal, we are able to improve the bound for secret key \( d < N^{0.292} \).

Let \( M \) be an \((m + 1)n \times (m + 1)n\) matrix. We index the columns such that the pair \((i, j)\) corresponds to the \((ni + j)\)th column of \( M \), where \( i = 0, \ldots, m \) and \( j = 1, \ldots, n \). Likewise we index the \((nk + l)\)th row by the pair \((k, l)\). We denote \( M(i, j, k, l) \) by the entry of \( M \) in the \((i, j)\)th column of the \((k, l)\)th row. Now we make the following definition.

**Definition 4.1** Let \( C, D, c_0, c_1, c_2, c_3, c_4, \zeta \) be real numbers with \( C, D, \zeta \geq 1 \). In order for a matrix \( M \) to be geometrically progressive with parameter \( C, D, c_0, c_1, c_2, c_3, c_4, \zeta \), for all \( i, k = 0, \ldots, m \) and \( j, l = 1, \ldots, n \), the following conditions must be met:

1. \[ |M(i, j, k, l)| \leq C \cdot D^{c_0 + c_1 + c_2 j + c_3 k + c_4 l}. \]
2. \[ M(k, l, k, l) = D^{c_0 + c_1 + c_2 j + c_3 k + c_4 l}. \]
3. \[ M(i, j, k, l) = 0 \text{ for } i > k \text{ or } j > l. \]
4: \( \zeta c_1 + c_3 \geq 0 \) and \( \zeta c_2 + c_4 \geq 0 \).

Now we will calculate a bound of the determinant of a \textit{geometrically progressive matrix} when a group of rows are removed.

**Theorem 4.1** Let \( M \) be an \((m+1)n \times (m+1)n\) geometrically progressive matrix with parameters \((C, D, c_0, c_1, c_2, c_3, c_4, \zeta)\) and let \( B \) be a real number. Let

\[
S_B = \{(k, l) \in \{0, ..., m\} \times \{1, ..., n\} | M(k, l, k, l) \leq B \text{ and } \omega = |S_B|\}.
\]

If a lattice \( L \) is defined by the rows \((k, l) \in S_B\) of \( M \), then

\[
\det(L) \leq ((m+1)n)^\frac{3}{2} (1 + C)^{w^2} \prod_{(k,l) \in S_B} M(k, l, k, l).
\]

The proof is very long and can be found in [3]. Now we want to show that the the sublattice of the Boneh-Durfee lattice generated by the \( y \)-shifts only are geometrically progressive. Recall that the \( y \)-shifts are the coefficients of the polynomials

\[
h_{l,k}(x, y) = y^l f^k(x, y)e^{m-k},
\]

for \( k = 0, ..., m \) and \( l = 1, ..., t \). Now we have the following.

**Lemma 4.1** For all positive integers \( m \) and \( t \), the matrix \( M_y \) generated by the \( y \)-shifts, is geometrically progressive with parameters \((m^{2m}, e, m, \frac{1}{2} + \delta, -\frac{1}{2}, -1, 1, 2)\).

**Proof:** For simplicity, we will take \( e = N^\alpha \) with \( \alpha = 1 \). Let \( (k, l) \) be given with \( k = 0, ..., m \) and \( l = 1, ..., t \). The row \((k, l)\) of \( M_y \) corresponds to the \( y \)-shift \( h_{l,k}(xX, yY) \). Notice

\[
h_{l,k}(xX, yY) = e^{m-k}y^l f^k(xX, yY) = \sum_{u=0}^{k} \sum_{v=0}^{u} c_{u,v} s^u y^{u+l},
\]

where

\[
c_{u,v} = \binom{k}{u} \binom{u}{v} (-1)^{k-u} e^{m-k} A^{u-v} X^u Y^{v+l}.
\]

Notice that for the pair \((i, j)\) corresponds to the coefficients of the \( x^i y^{i+j} \) in \( h_{l,k}(xX, yY) \); hence,
\[ M_y(i, j, k, l) = c_{i,i+j-l} = \binom{k}{i} \binom{i}{i+j-l} (-1)^{k-i} e^{m-k} A^{i-j} X^i Y^{i+j}. \]

Now notice that \( M_y(i, j, k, l) = c_{i,i+j-l} = 0 \) when \( i > k \) or \( j > l \); hence, condition 3 in Definition 4.1 is satisfied. Now we plug in the values \( X = e^\delta, Y = e^{1/2}, \) and assuming \( A < e \), we have

\[
|M_y(i, j, k, l)| = \left| \binom{k}{i} \binom{i}{i+j-l} (-1)^{k-i} e^{m-k} A^{i-j} X^i Y^{i+j} \right| \leq m^{2m} e^{m+(1/2+\delta)i-1/2j-k+l},
\]

which now satisfies condition 1. Now notice

\[
M_y(k, l, k, l) = e^{m+(1/2+\delta)i-1/2j-k+l},
\]

satisfying condition 2. Finally we have

\[
2(1/2 + \delta) + (-1) = 2\delta \geq 0 \text{ and } 2(-1/2) + 1 \geq 0,
\]

satisfying condition 4 and completing the proof.

**4.4 Bounding the Determinant of the New Lattice**

Now we shall apply GPM (geometrically progressive matrices) to the original lattice, calculate a new determinant and a new bound on \( d < N^{0.292} \). We start by setting \( t := (1 - 2\delta)k \). Now we perform Gaussian elimination on all the coefficients under the \( x \)-shifts. This would lead to Table 4.2 where \( \Lambda \) is a diagonal matrix and \( M_y' \) are the selected rows from \( M_y \). Notice for our new lattice \( L' \)

\[
\det(L') = \det(\Lambda) \cdot \det(M_y'),
\]

where \( L_y' \) is the set of integer combination of the rows of \( M_y' \). Now we need to establish a bound on \( \det(M_y') \) since we already know \( \det(\Lambda) \) from the previous section. The new lattice now has rank of \( w = (m+1)(m+2)/2 + w', \) where \( w' \) is the number of selected rows. Now let \( S \subseteq \{0,...,m\} \times \{1,...,t\} \) be the subset of indices such that \( M_y(k, l, k, l) \leq e^m \) for \((k, l) \in S\), so that \( w' = |S| \). Also, \((k, l) \in S\) only if

\[
e^{m+(\delta-1/2)k+1/2l} < e^m,
\]

21
Table 4.2: The Boneh-Durfee lattice after performing Gaussian elimination.

<table>
<thead>
<tr>
<th>x-shifts</th>
<th>y-shifts</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Lambda$</td>
<td>0</td>
</tr>
</tbody>
</table>

with $l \leq (1 - 2\delta)k$. In addition, every pair $(k, l)$ satisfies $l \leq (1 - 2\delta)k \leq t$, so $l \leq (1 - 2\delta)k$ if and only if $(k, l) \in S$. Hence,

$$w' = |S| = \sum_{k=0}^{m} [(1 - 2\delta)k] \geq \sum_{k=0}^{m} [(1 - 2\delta)k - 1] = (\frac{1}{2} - \delta)m^2 + o(m^2),$$

which implies

$$w = w' + (m + 1)(m + 2)/2 = (1 - \delta)m^2 + o(m^2).$$

By Theorem 4.1, we can establish a bound on the new lattice of all the $x$-shifts and the rows $(k, l) \in S$ $y$-shifts:

$$\det(L') \leq [(m + 1)(1 - 2\delta)m]^{w'/2}(1 + m^{2m})^{(w')^2} \prod_{(k,l)} M_y(k, l, k, l)$$

$$\leq [(m + 1)(1 - 2\delta)m]^{w'/2}(1 + m^{2m})^{(w')^2} \prod_{k=0}^{m} \prod_{l=0}^{[(1-2\delta)k]} e^{m+(\delta+\frac{1}{2})k+\frac{3}{8}l}$$

$$\leq [(m + 1)(1 - 2\delta)m]^{w'/2}(1 + m^{2m})^{(w')^2} e^{(\frac{5}{12} - \frac{2}{3})m^3 + o(m^3)},$$

where $[(m + 1)(1 - 2\delta)m]^{w'/2}(1 + m^{2m})^{(w')^2}$ is a function of only $\delta$ and not $e$. Hence, we can ignore it. Now we need to satisfy

$$\det(L') = \det(\Lambda) \cdot \det(L'_y) \leq e^{(\frac{5}{12} + \frac{2\delta}{3} - \frac{2\delta^2}{3})m^3 + o(m^3)} < e^{mw} = e^{(1-\delta)m^3 + o(m^3)},$$

which becomes

$$\left(-\frac{1}{6} + \frac{2\delta}{3} - \frac{2\delta^2}{3}\right)m^3 + o(m^3) < 0,$$

which implies

$$\delta < 1 - \frac{\sqrt{2}}{2} \approx 0.292.$$
For $\delta < 0.292$ and large $m$, LLL can produce two bivariate polynomials $r_1(x, y)$ and $r_2(x, y)$, such that $r_1(x_0, y_0) = r_2(x_0, y_0) = 0$. Boneh and Durfee make an heuristic that $r_1(x, y)$ and $r_2(x, y)$ will be algebraically independent. Hence, we can apply the resultant function, and obtain $f(y) = \text{Res}_x(r_1(x, y), r_2(x, y)) \in \mathbb{F}[y]$. Since $x_0$ and $y_0$ are roots in $r_1(x, y)$ and $r_2(x, y)$, $y_0$ must be a root in $f(y)$, which can be easily computed.

### 4.5 Boneh-Durfee Revisited

All of our experiments were performed under Linux on a 600 MHz Intel Pentium III processor. Experiments were carried out using the LLL implementation available in Victor Shoup’s NTL package [7]. All results in Tables 4.3 and 4.4 were successful in recovering $y_0 = -(p+q)$, confirming the heuristic that LLL will produce two algebraically independent polynomials.

#### Table 4.3: Our results of the Boneh-Durfee attack.

<table>
<thead>
<tr>
<th>Encryption</th>
<th>$d$</th>
<th>$\delta$</th>
<th>$m$</th>
<th>$t$</th>
<th>rank of lattice</th>
<th>our running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000 bits</td>
<td>520 bits</td>
<td>0.260</td>
<td>3</td>
<td>1</td>
<td>11</td>
<td>1 min</td>
</tr>
<tr>
<td>1000 bits</td>
<td>274 bits</td>
<td>0.269</td>
<td>5</td>
<td>2</td>
<td>25</td>
<td>6 min</td>
</tr>
<tr>
<td>1000 bits</td>
<td>278 bits</td>
<td>0.278</td>
<td>7</td>
<td>3</td>
<td>45</td>
<td>28 min</td>
</tr>
<tr>
<td>1000 bits</td>
<td>255 bits</td>
<td>0.279</td>
<td>7</td>
<td>3</td>
<td>45</td>
<td>28 min</td>
</tr>
</tbody>
</table>

#### Table 4.4: Boneh and Durfee’s original results of their attack.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$d$</th>
<th>$\delta$</th>
<th>$m$</th>
<th>$t$</th>
<th>rank of lattice</th>
<th>running time</th>
<th>advantage over Wiener</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 bits</td>
<td>280 bits</td>
<td>0.280</td>
<td>7</td>
<td>3</td>
<td>45</td>
<td>14 h</td>
<td>30 bits</td>
</tr>
<tr>
<td>2000 bits</td>
<td>550 bits</td>
<td>0.275</td>
<td>7</td>
<td>3</td>
<td>45</td>
<td>65 h</td>
<td>50 bits</td>
</tr>
<tr>
<td>4000 bits</td>
<td>1060 bits</td>
<td>0.265</td>
<td>5</td>
<td>2</td>
<td>25</td>
<td>14 h</td>
<td>60 bits</td>
</tr>
<tr>
<td>10 000 bits</td>
<td>2550 bits</td>
<td>0.255</td>
<td>3</td>
<td>1</td>
<td>11</td>
<td>90 min</td>
<td>50 bits</td>
</tr>
</tbody>
</table>
Chapter 5

HIGH ORDER BITS

In this chapter, we will improve the bound on $\delta$ by knowing the high-order bits of $p$. By knowing just $\frac{1}{100} \log_2 N$ high-order bits of $p$, one could break RSA for secret key $d < N^{0.30}$. We will also discuss ways to estimate $p + q$.

5.1 Boneh-Durfee Attack with High Bits Known

Suppose we know $\frac{1-2\beta}{2} \log_2 N$ high-order bits of $p$, for $0 \leq \beta < 0.5$. By division we know $\frac{1-2\beta}{2} \log_2 N$ high-order bits of $q$ as well. Hence, we can represent $p$ and $q$ by

$$p = P + p_0$$
$$q = Q + q_0,$$

where $P$ and $Q$ are known, and $p_0$ and $q_0$ are unknown. Assuming that $p$ and $q$ are balanced, from the small inverse problem [3] we have

$$y = -(p + q) = -(P + p_0 + Q + q_0) = -(B + y'),$$

where $y' = p_0 + q_0$, $|y'| < e^\beta$ and $B = P + Q$. Now the small inverse problem becomes

$$x(A + B + y') - 1 \equiv 0 \pmod{e}$$

where

$$|x| < e^\delta \text{ and } |y'| < e^\beta.$$
Table 5.1: The matrix spanned by \( g_{i,k} \) and \( h_{j,k} \) for \( k = 0, \ldots, 3 \), \( i = 0, \ldots, 3 - k \), and \( j = 0, 1 \).

<table>
<thead>
<tr>
<th>( e^3 )</th>
<th>( e^3 )</th>
<th>( e^3 )</th>
<th>( e^3 )</th>
<th>( e^3 )</th>
<th>( e^3 )</th>
<th>( e^3 )</th>
<th>( e^3 )</th>
<th>( e^3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x e^3 )</td>
<td>( X e^3 )</td>
<td>( X Y e^2 )</td>
<td>( X^2 Y e^2 )</td>
<td>( X^3 Y e^2 )</td>
<td>( X^3 Y^2 e^2 )</td>
<td>( X^3 Y^3 e^2 )</td>
<td>( X^3 Y^4 e^2 )</td>
<td></td>
</tr>
<tr>
<td>( x Y e^2 )</td>
<td>( X Y^2 e^2 )</td>
<td>( X^2 Y^2 e^2 )</td>
<td>( X^3 Y^2 e^2 )</td>
<td>( X^3 Y^3 e^2 )</td>
<td>( X^3 Y^4 e^2 )</td>
<td>( X^3 Y^5 e^2 )</td>
<td>( X^3 Y^6 e^2 )</td>
<td></td>
</tr>
<tr>
<td>( x Y^2 e )</td>
<td>( X Y^3 e )</td>
<td>( X^2 Y^3 e )</td>
<td>( X^3 Y^3 e )</td>
<td>( X^3 Y^4 e )</td>
<td>( X^3 Y^5 e )</td>
<td>( X^3 Y^6 e )</td>
<td>( X^3 Y^7 e )</td>
<td></td>
</tr>
<tr>
<td>( x Y^3 e )</td>
<td>( X Y^4 e )</td>
<td>( X^2 Y^4 e )</td>
<td>( X^3 Y^4 e )</td>
<td>( X^3 Y^5 e )</td>
<td>( X^3 Y^6 e )</td>
<td>( X^3 Y^7 e )</td>
<td>( X^3 Y^8 e )</td>
<td></td>
</tr>
<tr>
<td>( y f e )</td>
<td>( X Y f e )</td>
<td>( X^2 Y^2 f e )</td>
<td>( X^3 Y^2 f e )</td>
<td>( X^3 Y^3 f e )</td>
<td>( X^3 Y^4 f e )</td>
<td>( X^3 Y^5 f e )</td>
<td>( X^3 Y^6 f e )</td>
<td></td>
</tr>
<tr>
<td>( y f^2 e )</td>
<td>( X Y f^2 e )</td>
<td>( X^2 Y^2 f^2 e )</td>
<td>( X^3 Y^2 f^2 e )</td>
<td>( X^3 Y^3 f^2 e )</td>
<td>( X^3 Y^4 f^2 e )</td>
<td>( X^3 Y^5 f^2 e )</td>
<td>( X^3 Y^6 f^2 e )</td>
<td></td>
</tr>
<tr>
<td>( y f^3 e )</td>
<td>( X Y f^3 e )</td>
<td>( X^2 Y^2 f^3 e )</td>
<td>( X^3 Y^2 f^3 e )</td>
<td>( X^3 Y^3 f^3 e )</td>
<td>( X^3 Y^4 f^3 e )</td>
<td>( X^3 Y^5 f^3 e )</td>
<td>( X^3 Y^6 f^3 e )</td>
<td></td>
</tr>
</tbody>
</table>

By defining \( A' = A + B \), we have the following: given a polynomial \( f(x, y) = x(A' + y) - 1 \), find \((x_0, y_0)\) such that

\[
f(x_0, y_0) \equiv 0 \pmod{e}
\]

where

\[
|x_0| < e^\delta = X \quad \text{and} \quad |y_0| < e^\beta = Y.
\]

We define the following \( x \)-shifts and \( y \)-shifts:

\[
g_{i,k}(x, y) = x^i f^k(x, y)e^{m-k}
\]

and

\[
h_{j,k}(x, y) = y^j f^k(x, y)e^{m-k}.
\]

Given an integer \( m \), we build a lattice \( L \) spanned by the coefficient vectors of the polynomials for \( k = 0, \ldots, m \). For each \( k \), we use \( g_{i,k}(x X, y Y) \) for \( i = 0, \ldots, m - k \) and use \( h_{j,k}(x X, y Y) \) for \( j = 0, \ldots, t \) where \( t \) is minimized based on \( m \). This results in a full rank triangular matrix with \( w = (m + 1)(m + 2)/2 + t(m + 1) \) rows. The determinant of the lattice can be easily computed as the product of the diagonal entries. Table 5.1 is the lattice with \( m = 3 \) and \( t = 1 \). We denote \( \det_x \) as the determinant of the lattice due to the \( x \)-shifts, and \( \det_y \) the determinant due to the \( y \)-shifts:

\[
\det_x = e^{m(m+1)(m+2)/3} \cdot X^{m(m+1)(m+2)/3} \cdot Y^{m(m+1)(m+2)/6}
\]

and

\[
\det_y = e^{m(m+1)(m+2)/3} \cdot X^{m(m+1)(m+2)/3} \cdot Y^{m(m+1)(m+2)/6}
\]
\[
\text{det}_y = e^{tm(m+1)/2} \cdot X^{tm(m+1)/2} \cdot Y^{t(m+1)(m+t+1)/2}.
\]

Now when we substitute the values for \(X = e^\delta\) and \(Y = e^\beta\) we have
\[
\text{det}_x = e^{\frac{2+2\delta+\beta}{6}m^3 + o(m^3)}
\]

and
\[
\text{det}_y = e^{\frac{1+\delta+\beta}{2}tm^2 + t^2m\alpha/2 + o(m^2)}.
\]

Because of the Lemma 3.1 and Fact 3.2, we need to satisfy
\[
\text{det}(L) = \text{det}_x \cdot \text{det}_y < e^{mw}/\gamma,
\]

where \(\gamma = (w^{2w})^{w/2}\). Since \(w\) is only a function of \(\delta\) and not the public exponent \(e\), \(\gamma\) is negligible compared to \(e^{mw}\). Ignoring low order terms we have
\[
\text{det}(L) = e^{\frac{2+2\delta+\beta}{6}m^3 + \frac{1+\delta+\beta}{2}tm^2 + t^2m\alpha/2 + o(m^3)}.
\]

In order to satisfy \(\text{det}(L) < e^{mw}\), we have
\[
\frac{2+2\delta+\beta}{6}m^3 + \frac{1+\delta+\beta}{2}tm^2 + t^2m\beta/2 < \frac{1}{2}m^3 + tm^2,
\]

which leads to
\[
\frac{2\delta+\beta-1}{6}m^3 + \frac{\delta+\beta-1}{2}tm^2 + t^2m\beta/2 < 0.
\]

For every \(m\) the LHS is minimized at \(t = \frac{1-\delta-\beta}{2\beta}m\). We obtain new bound
\[
\delta < \frac{1}{3}(3 + \beta - 2\sqrt{\beta}\sqrt{3+\beta}).
\]

Hence for large enough \(m\), a certain \(\beta\) and \(\delta\) as in Table 5.2, we can find two bivariate polynomials \(r_1(x, y)\) and \(r_2(x, y) \in \mathbb{Z}[x, y]\) such that \(r_1(x_0, y_0) = r_2(x_0, y_0) = 0\). Heuristically the polynomials will be algebraically independent, hence \(h(y) = \text{Res}_x(r_1(x, y), r_2(x, y))\), where \(h(y)\) is a univariate polynomial with a root at \(y_0\).
Table 5.2: New bounds of $\delta$ for a given $\beta$.

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>$\delta$</th>
<th>$\beta$</th>
<th>$\delta$</th>
<th>$\beta$</th>
<th>$\delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>0.284</td>
<td>0.45</td>
<td>0.310</td>
<td>0.40</td>
<td>0.355</td>
</tr>
<tr>
<td>0.49</td>
<td>0.291</td>
<td>0.44</td>
<td>0.326</td>
<td>0.39</td>
<td>0.363</td>
</tr>
<tr>
<td>0.48</td>
<td>0.298</td>
<td>0.43</td>
<td>0.334</td>
<td>0.35</td>
<td>0.394</td>
</tr>
<tr>
<td>0.47</td>
<td>0.305</td>
<td>0.42</td>
<td>0.340</td>
<td>0.30</td>
<td>0.437</td>
</tr>
<tr>
<td>0.46</td>
<td>0.312</td>
<td>0.41</td>
<td>0.348</td>
<td>0.25</td>
<td>0.482</td>
</tr>
</tbody>
</table>

5.2 Geometrically Progressive Matrices

Here we will sharpen the relationship between $\beta$ and $\delta$ by throwing away vectors in the $y$-shift.

**Lemma 5.1** For all positive integers $m$ and $t$, the matrix $M_y$ generated by the $y$-shifts, is geometrically progressive with parameters $(m^{2m}, e, m, \beta + \delta, \beta - 1, -1, 1, \frac{1}{1-\beta})$.

**Proof:** For simplicity, we will take $e = N^\alpha$ with $\alpha = 1$. Let $(k, l)$ be given with $k = 0, ..., m$ and $l = 1, ..., t$. The row $(k, l)$ of $M_y$ corresponds to the $y$-shift $h_{l,k}(xX, yY)$. Notice

$$h_{l,k}(xX, yY) = e^{m-k}y^l f^k(xX, yY) = \sum_{u=0}^{m} \sum_{v=0}^{l} d_{u,v} s^{u}y^{v+l},$$

where

$$c_{u,v} = \begin{pmatrix} k \\ u \\ v \end{pmatrix} (-1)^{k-u} e^{m-k} A^{u-v} X^u Y^{v+l}. $$

Notice for the pair $(i, j)$ corresponds to the coefficients of the $x^i y^{i+j}$ in $h_{l,k}(xX, yY)$; hence,

$$M_y(i, j, k, l) = c_{i,i+j-l} = \begin{pmatrix} k \\ i \\ i+j-l \end{pmatrix} (-1)^{k-i} e^{m-k} A^{i-j} X^i Y^{i+j}. $$

Now notice $M_y(i, j, k, l) = c_{i,i+j-l} = 0$ when $i > k$ or $j > l$. Hence condition 3 in Definition 4.1 is satisfied. Now we plug in the values $X = e^\delta$, $Y = e^\beta$, and assuming $A < e$, we have

$$|M_y(i, j, k, l)| = \begin{pmatrix} k \\ i \\ i+j-l \end{pmatrix} (-1)^{k-i} e^{m-k} A^{i-j} X^i Y^{i+j} | \leq m^{2m} e^{m+(\beta+1)+((\beta-1)j-k)+l}. $$

Which now satisfies condition 1. Now notice
\[ M_y(k, l, k, l) = e^{m + (\beta + \delta)j + (\beta - 1)j-k+l}, \]

satisfying condition 2. Finally assuming \( 2\beta + \delta \geq 1 \), then

\[
\frac{1}{1-\beta}(\beta + \delta) + (-1) \geq 0 \quad \text{and} \quad \frac{1}{1-\beta}(\beta - 1) + 1 = 0 \geq 0.
\]

this satisfies condition 4 and completes the proof. ■

5.3 Bounding the Determinant of the New Lattice

Now we shall apply GPM to the original lattice, calculate a new determinant and a new bound on \( d \). Now let \( S \subseteq \{0, \ldots, m\} \times \{1, \ldots, t\} \) be the subset of indices such that \( M_y(k, l, k, l) \leq e^m \) for \( (k, l) \in S \), so that \( w' = |S| \). In addition \( (k, l) \in S \) only if

\[
e^{m+(\delta+\beta-1)k+\beta l} < e^m,
\]

with \( l \leq (1-2\delta)k \). Also, we have for every pair \( (k, l) \) satisfies \( l \leq (1-2\delta)k \leq t \), so \( l \leq (1-2\delta)k \) if and only if \( (k, l) \in S \). Hence

\[
w' = |S| = \sum_{k=0}^{m} [(1-2\delta)k] \geq \sum_{k=0}^{m} [(1-2\delta)k - 1] = (\frac{1}{2} - \delta)m^2 + o(m^2),
\]

which implies

\[
w = w' + (m + 1)(m + 2)/2 = (1 - \delta)m^2 + o(m^2).
\]

By Theorem 4.1, we can establish a bound on the new lattice of all the \( x \)-shifts and the rows \( (k, l) \in S \) \( y \)-shifts:

\[
\det(L'_y) \leq [(m + 1)(1 - 2\delta)m]^{w'/2}(1 + m^{2m})(w')^2 \prod_{(k, l) \in S} M_y(k, l, k, l) \\
\leq [(m + 1)(1 - 2\delta)m]^{w'/2}(1 + m^{2m})(w')^2 \prod_{k=0}^{m} \prod_{l=0}^{(1-2\delta)k} e^{m+(\delta+\beta-1)k+\beta l} \\
\leq [(m + 1)(1 - 2\delta)m]^{w'/2}(1 + m^{2m})(w')^2 e^{\left(\frac{2+\beta}{1-2\delta} - \frac{\delta}{1-2\delta} + 2\frac{\beta+2\delta}{1-2\delta}\right)m^3 + o(m^3)},
\]

where \([(m + 1)(1 - 2\delta)m]^{w'/2}(1 + m^{2m})(w')^2\) is a function of only \( \delta \) and not \( e \). Hence we can ignore it. Now we need to satisfy
\[
\det(L') = \det(A) \cdot \det(L'_y) \leq e^{(2+2\beta+\delta) \cdot \frac{m^3 + (1+3\beta - \frac{4\beta}{3} + 2\beta + 2\delta^2) m^3 + o(m^3)}{6}} < e^{m w} = e^{(1-\delta)m^3 + o(m^3)},
\]

which becomes

\[
\left(\frac{4\beta - 3}{6} + 3\delta + \left(\frac{2\beta - 2}{3}\right) \delta^2\right) m^3 + o(m^3) < 0,
\]

which implies

\[
\delta < \frac{2\beta + \sqrt{1-\beta} - 2}{2(\beta - 1)}.
\]

Notice that as \( \beta > 0.40 \) in Table 5.3, and knowing more than \( \frac{1}{25} \log_2 N \) high-order bits of \( p \), applying GPM gives a poorer bound for \( \delta \). Assuming \( \beta, \delta < 0.50 \), notice

\[
1 < 2 + \sqrt{1-\beta} + (2\beta - 1)\delta < 2\beta + \delta.
\]

satisfying condition 4 in Definition 4.1.

### Table 5.3: New bounds of \( \delta \) for a given \( \beta \).

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>( \delta )</th>
<th>( \beta )</th>
<th>( \delta )</th>
<th>( \beta )</th>
<th>( \delta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50</td>
<td>0.292</td>
<td>0.45</td>
<td>0.326</td>
<td>0.40</td>
<td>0.354</td>
</tr>
<tr>
<td>0.49</td>
<td>0.300</td>
<td>0.44</td>
<td>0.332</td>
<td>0.39</td>
<td>0.360</td>
</tr>
<tr>
<td>0.48</td>
<td>0.307</td>
<td>0.43</td>
<td>0.338</td>
<td>0.35</td>
<td>0.380</td>
</tr>
<tr>
<td>0.47</td>
<td>0.313</td>
<td>0.42</td>
<td>0.343</td>
<td>0.30</td>
<td>0.403</td>
</tr>
<tr>
<td>0.46</td>
<td>0.320</td>
<td>0.41</td>
<td>0.349</td>
<td>0.25</td>
<td>0.423</td>
</tr>
</tbody>
</table>

### 5.4 Methods on Obtaining High Order Bits

It is relatively hard to obtain the high-order bits of \( p \). However, there are methods in estimating \( p + q \). If \( p \) and \( q \) have small differences, then \( p + q \approx 2\sqrt{N} \). This is very similar to Weger’s work [8]. Recall that

\[
\frac{\sqrt{N}}{2} < q < p < 2\sqrt{N};
\]
hence,

\[2\sqrt{N} < q + p < \frac{5\sqrt{N}}{2}\]

since \(N = pq\). When \(p\) and \(q\) have large differences, \(p + q \approx 2.5\sqrt{N}\). For \(k \in (2.0, 2.5) \subset \mathbb{R}\), guessing \(p + q = k\sqrt{N}\) is much easier and more accurate rather than guessing the high order bits of \(p\).

### 5.5 Experiments

All experiments were performed under Linux on a 600-MHz Intel Pentium III processor. Experiments were carried out using the LLL implementation available in Victor Shoup’s NTL package.

All experiments produced two algebraically independent polynomials \(r_1(x, y)\) and \(r_2(x, y)\) with \(x_0\) and \(y_0\) as roots. We recovered \(y_0\) by calculating the roots of the resultant \(\text{Res}_x(r_1(x, y), r_2(x, y)) = (y - y_0)f(y) \in F[y]\). \(d \in \lbrack \frac{3}{4}N^\delta, N^\delta \rbrack\) was chosen at random. We have broken down our analysis based on the rank of lattice we have chosen. See Tables 5.4 - 5.8.

#### Table 5.4: Cryptanalysis of 1000-bit encryption with lattice rank = 11.

<table>
<thead>
<tr>
<th>(N)</th>
<th>(d)</th>
<th>(\delta)</th>
<th>(m)</th>
<th>(t)</th>
<th>(\beta)</th>
<th>(\frac{1-2\beta}{2} \log_2 N)</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 bits</td>
<td>272 bits</td>
<td>0.272</td>
<td>3</td>
<td>1</td>
<td>0.49</td>
<td>10 bits</td>
<td>6 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>277 bits</td>
<td>0.277</td>
<td>3</td>
<td>1</td>
<td>0.48</td>
<td>20 bits</td>
<td>6 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>282 bits</td>
<td>0.282</td>
<td>3</td>
<td>1</td>
<td>0.47</td>
<td>30 bits</td>
<td>6 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>287 bits</td>
<td>0.287</td>
<td>3</td>
<td>1</td>
<td>0.46</td>
<td>40 bits</td>
<td>6 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>295 bits</td>
<td>0.295</td>
<td>3</td>
<td>1</td>
<td>0.45</td>
<td>50 bits</td>
<td>6 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>301 bits</td>
<td>0.301</td>
<td>3</td>
<td>1</td>
<td>0.44</td>
<td>60 bits</td>
<td>6 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>312 bits</td>
<td>0.312</td>
<td>3</td>
<td>1</td>
<td>0.42</td>
<td>80 bits</td>
<td>6 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>325 bits</td>
<td>0.325</td>
<td>3</td>
<td>1</td>
<td>0.40</td>
<td>100 bits</td>
<td>6 s</td>
</tr>
</tbody>
</table>
Table 5.5: Cryptanalysis of 2000-bit encryption with lattice rank = 11.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$d$</th>
<th>$\delta$</th>
<th>$m$</th>
<th>$t$</th>
<th>$\beta$</th>
<th>$\frac{1-2\beta}{2} \log_2 N$</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000 bits</td>
<td>538 bits</td>
<td>0.269</td>
<td>3</td>
<td>1</td>
<td>0.49</td>
<td>20 bits</td>
<td>40 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>552 bits</td>
<td>0.276</td>
<td>3</td>
<td>1</td>
<td>0.48</td>
<td>40 bits</td>
<td>40 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>562 bits</td>
<td>0.281</td>
<td>3</td>
<td>1</td>
<td>0.47</td>
<td>60 bits</td>
<td>40 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>574 bits</td>
<td>0.287</td>
<td>3</td>
<td>1</td>
<td>0.46</td>
<td>80 bits</td>
<td>40 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>588 bits</td>
<td>0.294</td>
<td>3</td>
<td>1</td>
<td>0.45</td>
<td>100 bits</td>
<td>40 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>600 bits</td>
<td>0.300</td>
<td>3</td>
<td>1</td>
<td>0.44</td>
<td>120 bits</td>
<td>40 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>622 bits</td>
<td>0.311</td>
<td>3</td>
<td>1</td>
<td>0.42</td>
<td>160 bits</td>
<td>40 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>648 bits</td>
<td>0.324</td>
<td>3</td>
<td>1</td>
<td>0.40</td>
<td>200 bits</td>
<td>40 s</td>
</tr>
</tbody>
</table>

Table 5.6: Cryptanalysis of 4000-bit encryption with lattice rank = 11.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$d$</th>
<th>$\delta$</th>
<th>$m$</th>
<th>$t$</th>
<th>$\beta$</th>
<th>$\frac{1-2\beta}{2} \log_2 N$</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000 bits</td>
<td>1072 bits</td>
<td>0.268</td>
<td>3</td>
<td>1</td>
<td>0.49</td>
<td>40 bits</td>
<td>3 min</td>
</tr>
<tr>
<td>4000 bits</td>
<td>1096 bits</td>
<td>0.274</td>
<td>3</td>
<td>1</td>
<td>0.48</td>
<td>80 bits</td>
<td>3 min</td>
</tr>
<tr>
<td>4000 bits</td>
<td>1124 bits</td>
<td>0.281</td>
<td>3</td>
<td>1</td>
<td>0.47</td>
<td>120 bits</td>
<td>3 min</td>
</tr>
<tr>
<td>4000 bits</td>
<td>1148 bits</td>
<td>0.287</td>
<td>3</td>
<td>1</td>
<td>0.46</td>
<td>160 bits</td>
<td>3 min</td>
</tr>
<tr>
<td>4000 bits</td>
<td>1176 bits</td>
<td>0.294</td>
<td>3</td>
<td>1</td>
<td>0.45</td>
<td>200 bits</td>
<td>3 min</td>
</tr>
<tr>
<td>4000 bits</td>
<td>1200 bits</td>
<td>0.300</td>
<td>3</td>
<td>1</td>
<td>0.44</td>
<td>240 bits</td>
<td>3 min</td>
</tr>
<tr>
<td>4000 bits</td>
<td>1244 bits</td>
<td>0.311</td>
<td>3</td>
<td>1</td>
<td>0.42</td>
<td>320 bits</td>
<td>3 min</td>
</tr>
<tr>
<td>4000 bits</td>
<td>1296 bits</td>
<td>0.324</td>
<td>3</td>
<td>1</td>
<td>0.40</td>
<td>400 bits</td>
<td>3 min</td>
</tr>
</tbody>
</table>

Table 5.7: Cryptanalysis of 1000-bit encryption with lattice rank = 25.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$d$</th>
<th>$\delta$</th>
<th>$m$</th>
<th>$t$</th>
<th>$\beta$</th>
<th>$\frac{1-2\beta}{2} \log_2 N$</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 bits</td>
<td>280 bits</td>
<td>0.280</td>
<td>5</td>
<td>2</td>
<td>0.49</td>
<td>10 bits</td>
<td>8 min</td>
</tr>
<tr>
<td>1000 bits</td>
<td>288 bits</td>
<td>0.288</td>
<td>5</td>
<td>2</td>
<td>0.48</td>
<td>20 bits</td>
<td>8 min</td>
</tr>
<tr>
<td>1000 bits</td>
<td>292 bits</td>
<td>0.292</td>
<td>5</td>
<td>2</td>
<td>0.47</td>
<td>30 bits</td>
<td>8 mins</td>
</tr>
<tr>
<td>1000 bits</td>
<td>298 bits</td>
<td>0.298</td>
<td>5</td>
<td>2</td>
<td>0.46</td>
<td>40 bits</td>
<td>8 min</td>
</tr>
<tr>
<td>1000 bits</td>
<td>302 bits</td>
<td>0.302</td>
<td>5</td>
<td>2</td>
<td>0.45</td>
<td>50 bits</td>
<td>8 min</td>
</tr>
<tr>
<td>1000 bits</td>
<td>307 bits</td>
<td>0.307</td>
<td>5</td>
<td>2</td>
<td>0.44</td>
<td>60 bits</td>
<td>8 min</td>
</tr>
<tr>
<td>1000 bits</td>
<td>319 bits</td>
<td>0.319</td>
<td>5</td>
<td>2</td>
<td>0.42</td>
<td>80 bits</td>
<td>8 min</td>
</tr>
<tr>
<td>1000 bits</td>
<td>333 bits</td>
<td>0.333</td>
<td>5</td>
<td>2</td>
<td>0.40</td>
<td>100 bits</td>
<td>8 min</td>
</tr>
</tbody>
</table>
Table 5.8: Cryptanalysis of 1000-bit encryption with lattice rank = 45.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$d$</th>
<th>$\delta$</th>
<th>$m$</th>
<th>$t$</th>
<th>$\beta$</th>
<th>$\frac{1-2\beta}{2} \log_2 N$</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 bits</td>
<td>285 bits</td>
<td>0.285</td>
<td>7</td>
<td>3</td>
<td>0.49</td>
<td>10 bits</td>
<td>3 h</td>
</tr>
<tr>
<td>1000 bits</td>
<td>291 bits</td>
<td>0.291</td>
<td>7</td>
<td>3</td>
<td>0.48</td>
<td>20 bits</td>
<td>3 h</td>
</tr>
<tr>
<td>1000 bits</td>
<td>298 bits</td>
<td>0.298</td>
<td>7</td>
<td>3</td>
<td>0.47</td>
<td>30 bits</td>
<td>3 h</td>
</tr>
<tr>
<td>1000 bits</td>
<td>304 bits</td>
<td>0.304</td>
<td>7</td>
<td>3</td>
<td>0.46</td>
<td>40 bits</td>
<td>3 h</td>
</tr>
<tr>
<td>1000 bits</td>
<td>308 bits</td>
<td>0.308</td>
<td>7</td>
<td>3</td>
<td>0.45</td>
<td>50 bits</td>
<td>3 h</td>
</tr>
<tr>
<td>1000 bits</td>
<td>317 bits</td>
<td>0.317</td>
<td>7</td>
<td>3</td>
<td>0.44</td>
<td>60 bits</td>
<td>3 h</td>
</tr>
<tr>
<td>1000 bits</td>
<td>327 bits</td>
<td>0.327</td>
<td>7</td>
<td>3</td>
<td>0.42</td>
<td>80 bits</td>
<td>3 h</td>
</tr>
<tr>
<td>1000 bits</td>
<td>339 bits</td>
<td>0.339</td>
<td>7</td>
<td>3</td>
<td>0.40</td>
<td>100 bits</td>
<td>3 h</td>
</tr>
</tbody>
</table>

5.6 Conclusion

This was a very simple extension of the Boneh-Durfee attack. We were successful in obtaining algebraically independent polynomials because our lattice is very similar to the Boneh-Durfee’s lattice [3]. The real magic of the Boneh-Durfee attack is that it produced algebraically independent polynomials. In the next chapter we will see that it is very hard to produce algebraically independent polynomials using LLL, and estimating $p + q$ might be the best way to improve lattice attacks on RSA with short secret key $d$. 
Chapter 6

IMPROVING BOUNDS

In this chapter we will stress the importance of finding the necessary and sufficient conditions for reduced vectors to be algebraically independent. Here we propose an attack on RSA for secret key $d < N^{0.365}$. We also propose an attack on the factoring with a hint problem by knowing $\frac{1}{6} \log_2 N$ high-order bits of $p$. However, the attacks do not work yet. In all of our experiments, the LLL algorithm did not produce algebraically independent polynomials.

6.1 A New Equation

We continue the attack on RSA where the public exponent $e = N^\alpha$ for $\alpha \approx 1$. We go back to the small inverse problem, and create a lattice to generate a new equation to solve. Recall the small inverse problem:

$$k(A + s) - 1 \equiv 0 \pmod{e}$$

where

$$|s| < e^{0.5} \text{ and } |k| < e^{0.5}.$$ 

Now given a polynomial $f(x, y) = x(A + y) - 1$, find $(x_0, y_0)$ such that

$$f(x_0, y_0) \equiv 0 \pmod{e}$$

where

$$|x_0| < e^{0.5} = X \text{ and } |y_0| < e^{0.5} = Y.$$
We define the following:

\[ g_i(x, y) = x^{1-i} f^i(x, y) e^{1-i}. \]

For all \( i \in \{0, 1\} \), we span the lattice \( L_p \) with rows of the coefficients of \( g_i(x, y) \):

\[
L_p = \begin{pmatrix} eX \\ -1 & AX & XY \end{pmatrix}.
\]

Notice \( \det(L_p) = eX \sqrt{1 + X^2Y^2} \approx eX^2Y \). Now when we apply LLL to \( L_p \), we get a new basis of reduced vectors \( r_1 \) and \( r_2 \). We also obtain the transformation matrix \( T \). For

\[
L_p \rightarrow \text{LLL} \rightarrow R = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix},
\]

we have

\[
TL_p = R
\]

and

\[
T = \begin{pmatrix} C_{1,1} & C_{1,2} \\ C_{2,1} & C_{2,2} \end{pmatrix},
\]

where \( C_{i,j} \in \mathbb{Z} \) for \( i, j \in \{1, 2\} \). We now have \( r_1(x, y) = C_{1,1}exe + C_{1,2}f(x, y) = c_1e \) and \( r_2(x, y) = C_{2,1}exe + C_{2,2}f(x, y) = c_2e \) for some integer \( c_1 \) and \( c_2 \). By Fact 3.2 and Lemma 3.1 we have

\[
|r_1(x_0, y_0)| = |C_{1,1}exe_0 - C_{1,2}ed| = |c_1e| \leq \sqrt{2} |r_1(xX, yY)| \leq 2^{1/2} \det(L_p)^{1/2}.
\]

By ignoring small-order terms, we have \( |c_1| < e^{\delta - 0.25} \). To bound our second reduced vector \( r_2 \), we recognize that the lower bound of the shortest vector in \( L_p \) is \( \lambda_{min}^* \approx XY \). By Fact 3.3 we have

\[
||r_2|| < \max \left( 2\sqrt{eX^2Y}, \frac{2eX^2Y}{X^2} \right) = \max \left( 2\sqrt{eX^2Y}, 2eX \right) = 2eX,
\]

since \( Y < e^{0.5} \). However for the lattice \( L_p \), we believe this is not an accurate bound since:

\[
|r_2(x_0, y_0)| = |C_{2,1}exe_0 - C_{2,2}ed| = |c_2e| < 2eX
\]
which implies

$$|c_2| < e^\delta.$$  

This is an obvious statement, indicating that $r_2$ will be shorter than the vector $\langle -1, AX, XY \rangle$. Hence we make a heuristic claim that $|c_1| \approx |c_2| < e^{\delta-0.25}$.

**Heuristic 6.1** Given the lattice $L_p$, the LLL algorithm will generate two relatively short vectors $r_1$ and $r_2$, such that

$$||r_1|| \approx ||r_2|| \leq 2^{3/2} \det(L_p)^{1/2}.$$  

The heuristic implies $|c_1| \approx |c_2| \leq e^{\delta-0.25}$. Now notice that we have

$$C_{1,1}x_0 - C_{1,2}d = c_1$$

and

$$C_{2,1}x_0 - C_{2,2}d = c_2.$$  

Boneh and Durfee [3] showed that RSA is insecure for secret key $d = N^\delta < N^{0.292}$. Hence, we can assume that $\delta > 0.292$, which implies $|x_0| > N^{0.292} \approx e^{0.292}$. Now we can assume that $|C_{1,1}| \approx |C_{1,2}|$ and $|C_{2,1}| \approx |C_{2,2}|$. By manipulating the equations above we have:

$$(C_{1,1}C_{2,2} - C_{1,2}C_{2,1})d = C_{2,1}c_1 - C_{1,1}c_2.$$  

Now remember that the determinant of the matrix $L_p$ equals the determinant of the matrix $R$. We have $D(TL_p) = D(R)$ which implies $D(T)D(L_p) = D(R)$, where $D(\cdot)$ denotes the determinant of a matrix. Finally we have $D(T) = C_{1,1}C_{2,2} - C_{1,2}C_{2,1} = 1$. Hence,

$$d = C_{2,1}c_1 - C_{1,1}c_2.$$  

Since $|c_1|, |c_2| < e^{\delta-0.25}$, this implies that $\exists C_{i,j} > e^{0.25}$ for $(i, j) \in \{(1, 1), (2, 1)\}$. For simplicity we will set $|C_{1,1}| > e^{0.25}$. Now we have the following relationship:

$$-1 + Ak + ks = -e(C_{2,1}c_1 - C_{1,1}c_2)$$

35
and

\[-1 + Ak + eC_{2,1}c_1 + ks = C_{1,1}ec_2 \equiv 0 \pmod{C_{1,1}e}.\]

Where integers $A$, $C_{2,1}$, $C_{1,1}$ and $e$ are known and integers $k$, $c_1$ and $s$ are unknown. We will call this the modified small inverse problem.

### 6.2 Solving the Modified Small Inverse Problem

Now by setting $B = eC_{2,1}$ and $u = C_{1,1}e > e^{1.25}$, we have the following: given a polynomial $f(x, y, z) := Ax + Bz + xy - 1$, find $(x_0, y_0, z_0)$ such that

\[f(x_0, y_0, z_0) \equiv 0 \pmod{u}\]

where

\[|x_0| < u^{44/5} = X, \quad |y_0| < u^{0.4} = Y, \quad \text{and} \quad |z_0| < u^{44-1/5} = Z.\]

Now remember we already have one equation for $x$, $y$ and $z$. From the first lattice $L_p$, we have

\[r_1(x, y) = C_{1,1}ex + C_{1,2}f(x, y) = ec_1.\]

We define

\[p(x, y, z) = C_{1,1}ex + C_{1,2}f(x, y) - ez,\]

where $p(x_0, y_0, z_0) = 0$. For a given integer $m$, we define the polynomials

\[g_{i,j,k}(x, y, z) = x^iz^jf^k(x, y, z)u^{m-k}\]

and

\[h_{a,b,k}(x, y, z) = y^az^bf^k(x, y, z)u^{m-k}.\]

We refer the polynomials $g_{i,j,k}$ as the $x$-shifts and $h_{a,b,k}$ as the $y$-shifts. Now we span a lattice with coefficients of the polynomials $g_{i,j,k}(xX, yY, zZ)$ and $h_{a,b,k}(xX, yY, zZ)$, calculate the determinant and bound the reduced vectors $r_1$ and $r_2$. Making a second heuristic that polynomials
Substituting the values $t = 1$. Considering the $h$-term since it is equivalent to the product of the diagonal entries. Table 6.1 is the lattice for $m = 2$, $k = 0, 1, 2$, $i = 0, ..., 2 - k$, $j = 1, ..., 2 - k - i$, $a = 1$ and $b = 0, ..., 2 - k$.

For a given integer $m$, we create a lattice spanned by the coefficients of $g_{i,j,k}(xX, yY, zZ)$ and $h_{a,b,k}(xX, yY, zZ)$. We span the $x$-shifts for $k = 0, ..., m$, $i = 0, ..., m - k$ and $j = 0, ..., m - k - i$. We span the $y$-shifts for $k = 0, ..., m$, $a = 1, ..., t$ and $b = 0, ..., m - k$, where we will determine $t$ later. Our lattice will be at full rank, and we can easily calculate the determinant of the lattice since it is equivalent to the product of the diagonal entries. Table 6.1 is the lattice for $m = 2$ and $t = 1$. Considering the $x$-shifts only, we create $(6 + 11m + 6m^2 + m^3)/6$ rows and:

$$
\det_x = u^{m(1+m)(2+m)(3+m)/8} \cdot X^{m(1+m)(2+m)(3+m)/12} \cdot Y^{m(1+m)(2+m)(3+m)/24} \cdot Z^{m(1+m)(2+m)(3+m)/24}.
$$

Substituting the values $X = u^{4s/5}$, $Y = u^{0.4}$ and $Z = u^{4d/5}$ we have:

$$
\det_x = u^{(\frac{12s}{17} + \frac{1}{m}(\frac{2}{5} + d + \frac{4}{17})m^3 + o(m^4)}.
$$

Now in order to satisfy Lemma 3.1, we have $\det_x < u^{m^4/6 + o(m^4)}$, which implies $\delta < 0.333$. Here we already have an improvement over the original bound $\delta < 0.292$ [3]. Now we will attempt to improve this bound by including the $y$-shifts. We calculate the determinant of the submatrix which is generated by the $y$-shifts as the product of the the diagonals:

$$
\det_y = u^{tm(1+m)(2+m)/3} \cdot X^{tm(1+m)(2+m)/6} \cdot Y^{(1+m)(2+m)(3t(1+t)/2t+m)/6} \cdot Z^{tm(1+m)(2+m)/6}.
$$
Substituting the values of $X$, $Y$, and $Z$:

$$\det_y = u^m(\frac{11t}{10} + \frac{4t\delta}{15}) + \frac{m^2t^2}{10} + o(m^3t) + o(m^2t^2).$$

From the $y$-shifts, we generate $t(m + 1)(m + 2)/2$ rows. We now have a lattice $L$ of dimension $w = (6 + 11m + 6m^2 + m^3)/6 + t(m + 1)(m + 2)/2$ and we need to satisfy the determinant such that

$$\det(L) = \det_x \cdot \det_y < u^{mw}/\gamma$$

where $\gamma = (w2^w)^{w/2}$. Since $\gamma$ is a function of the dimension of the lattice and not $u$, it is insignificant compared to $u^{mw}$. Hence we can ignore $\gamma$, and by ignoring other low-order terms, we need to satisfy

$$\det(L) = \det_x \cdot \det_y = u^{(\frac{17}{120} + \frac{1}{30}(-\frac{1}{4} + \delta) + \frac{\delta}{15})m^4 + m^3(\frac{11t}{10} + \frac{4t\delta}{15}) + \frac{m^2t^2}{10} < m^4/6 + tm^3/2},$$

This implies

$$\left(\frac{17}{120} + \frac{1}{30}(-\frac{1}{4} + \delta) + \frac{\delta}{15}\right)m^4 + m^3(\frac{11t}{10} + \frac{4t\delta}{15}) + \frac{m^2t^2}{10} < m^4/6 + tm^3/2,$$

which leads to

$$\frac{-1}{30}m^4 + \frac{1}{10}\delta m^4 - \frac{2}{15}m^3t + \frac{4}{15}\delta m^3t + \frac{1}{10}m^2t^2 < 0.$$

For all integers $m$, the LHS will be minimized for $t = \frac{2-4\delta}{3}m$. Plugging $t$ in, we have

$$-7 + 25\delta - 16\delta^2 < 0,$$

which give us

$$\delta < \frac{1}{32}(25 - \sqrt{177}) \approx 0.365.$$

Hence for lattice $L$, large $m$, and $\delta < 0.365$ the LLL algorithm will produce two three-variable polynomials such that $r_1(x_0, y_0, z_0) = r_2(x_0, y_0, z_0) = 0$. 

38
6.3 Factoring With a Hint

In 1985, Rivest and Shamir [9] showed that if one knows $\frac{1}{3}\log_2 N$ high-order bits of $p$, where $N = pq$ and $N$ is known, then one could factor $N$. The latest improvement was in 1997 where Coppersmith [10] showed that if one knows only $\frac{1}{4}\log_2 N$ high-order bits of $p$, then one could factor $N$ with lattice attacks. Now suppose we know the $\frac{1-2\alpha}{2}\log_2 N$ bits of $p$. By division we know the high-order $\frac{1-2\alpha}{2}\log_2 N$ bits of $q$ as well. Hence we can represent $p$ and $q$ by

$$
\begin{align*}
    p &= P + p_0 \\
    q &= Q + q_0,
\end{align*}
$$

where $P$ and $Q$ are known, and $p_0$ and $q_0$ are unknown. Assuming the $p$ and $q$ are balanced, we have the following equation:

$$(P + p_0)(Q + q_0) = PQ + Qp_0 + Pq_0 + p_0q_0 = N.$$

We set $x_0 = p_0$ and $y_0 = q_0$. Now we have

$$
Qx_0 + Py_0 + x_0y_0 + PQ = N \equiv 0 \pmod{N}
$$

where

$$
|x_0| \text{ and } |y_0| < N^\alpha < N^{0.5}.
$$

Now we have the following: given a polynomial $f(x, y) = Qx + Py + xy + PQ$, find $(x_0, y_0)$ such that

$$
f(x_0, y_0) \equiv 0 \pmod{N}
$$

where

$$
|x_0| = |y_0| < N^\alpha = X = Y.
$$

We define the following $x$-shifts and $y$-shifts:

$$
g_{i,k}(x, y) = x^i f^k(x, y) N^{m-k}
$$
and

\[ h_{j,k}(x, y) = y^j f^k(x, y) N^{m-k}. \]

Given an integer \( m \), we build a lattice \( L \) spanned by the coefficients of the polynomials for \( k = 0, ..., m \). For each \( k \) we use \( g_{i,k}(xX, yY) \) for \( i = 0, ..., m-k \) and \( h_{j,k}(xX, yY) \) for \( j = 1, ..., m-k \). This results in a full rank triangular matrix with \( (m+1)(m+2)/2 + m(m+1)/2 \) rows. The determinant of the lattice \( L \) can be easily computed as the product of the diagonal entries

\[ \det(L) = N^{m(m+1)(m+2)/3 + m(m+1)(2m+1)/6} \cdot X^{m(m+1)^2/2} \cdot Y^{m(m+1)^2/2}. \]

Now when we substitute the values for \( X = Y = N^\alpha \), we have

\[ \det(L) = N^{2/3m^3 + \alpha m^3 + o(m^3)}. \]

To apply Lemma 3.1 and Fact 3.2, we need to satisfy

\[ \det(L) < N^{mw}/\gamma, \tag{6.1} \]

where \( \gamma = (w2^w)^{w/2} \). Since \( \gamma \) is negligible compared to \( N^{mw} \), we can ignore it. Hence, by ignoring low-order terms we have

\[ \det(L) = N^{2/3m^3 + \alpha m^3 + o(m^3)}. \]

To satisfy \( \det(L) < N^{mw} \) we have

\[ 2/3m^3 + \alpha m^3 < m^3, \]

which leads to

\[ \alpha < 1/3. \]

This implies we need to know the high-order \( \frac{1}{6} \log_2 N \) high order bits of \( p \). Hence, for large enough \( m \), and \( \alpha < 1/3 \), we can find a bivariate polynomial \( r_1 \in \mathbb{Z}[x, y] \) such that \( r_1(x_0, y_0) = 0 \) over the integers. Table 6.2 is the lattice for \( m = 2 \). Now remember we already have one bivariate polynomial with \( x_0 \) and \( y_0 \) as roots

\[ PQ + Qx + Py + xy - N = 0. \]

If \( r_1 \) is algebraically independent of the equation above, we can solve for \( x_0 \) and \( y_0 \).
Table 6.2: The matrix spanned by $g_{i,k}$ and $h_{j,k}$ for $k = 0, \ldots, 2$, $i = 0, \ldots, 2 - k$, and $j = 1, \ldots, 2 - k$.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$N^2$</th>
<th>$xN^2$</th>
<th>$yN^2$</th>
<th>$fN$</th>
<th>$f^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N^2$</td>
<td>$N^2$</td>
<td>$XN^2$</td>
<td>$XN^2$</td>
<td>$YN^2$</td>
<td>$YN^2$</td>
</tr>
<tr>
<td>$xN^2$</td>
<td>$x^2N^2$</td>
<td>$X^2N^2$</td>
<td>$X^2N^2$</td>
<td>$fN$</td>
<td>$f^2$</td>
</tr>
<tr>
<td>$yN^2$</td>
<td>$y^2N^2$</td>
<td>$XYN$</td>
<td>$X^2YN$</td>
<td>$fN$</td>
<td>$f^2$</td>
</tr>
<tr>
<td>$xN$</td>
<td>$x^2N$</td>
<td>$XN$</td>
<td>$X^2N$</td>
<td>$fN$</td>
<td>$f^2$</td>
</tr>
<tr>
<td>$yN$</td>
<td>$y^2N$</td>
<td>$YN$</td>
<td>$Y^2N$</td>
<td>$fN$</td>
<td>$f^2$</td>
</tr>
<tr>
<td>$f$</td>
<td>$f^2$</td>
<td>$XY$</td>
<td>$XY^2$</td>
<td>$f$</td>
<td>$f^2$</td>
</tr>
</tbody>
</table>

6.4 Experiments

6.4.1 The heuristic

All experiments were performed on Linux with a 600-MHz Intel Pentium III processor. Experiments were carried out using the LLL implementation available in Victor Shoup’s NTL package [7]. The integer $d \in [\frac{3}{4}N^5, N^8]$ was chosen at random. Here we set the upper bound for $\text{Max}(c_1, c_2) < e^\theta$. All of our experiments confirmed our heuristic that $c_1$ and $c_2 \leq e^0 = e^{\delta - 0.25}$. Results are in Table 6.3.

Table 6.3: Results of lattice reduction on $L_p$ and size of $c_1$ and $c_2$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$d$</th>
<th>$\delta$</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$\theta$</th>
<th>Running time to reduce $L_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 bits</td>
<td>300 bits</td>
<td>0.30</td>
<td>50 bits</td>
<td>50 bits</td>
<td>0.05</td>
<td>1 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>400 bits</td>
<td>0.40</td>
<td>150 bits</td>
<td>150 bits</td>
<td>0.15</td>
<td>1 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>500 bits</td>
<td>0.50</td>
<td>250 bits</td>
<td>250 bits</td>
<td>0.25</td>
<td>1 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>600 bits</td>
<td>0.60</td>
<td>350 bits</td>
<td>350 bits</td>
<td>0.35</td>
<td>1 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>1000 bits</td>
<td>1.00</td>
<td>750 bits</td>
<td>750 bits</td>
<td>0.75</td>
<td>1 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>600 bits</td>
<td>0.30</td>
<td>100 bits</td>
<td>100 bits</td>
<td>0.05</td>
<td>16 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>800 bits</td>
<td>0.40</td>
<td>300 bits</td>
<td>300 bits</td>
<td>0.15</td>
<td>16 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>1000 bits</td>
<td>0.50</td>
<td>500 bits</td>
<td>500 bits</td>
<td>0.25</td>
<td>16 s</td>
</tr>
<tr>
<td>2000 bits</td>
<td>2000 bits</td>
<td>1.00</td>
<td>1500 bits</td>
<td>1500 bits</td>
<td>0.75</td>
<td>16 s</td>
</tr>
<tr>
<td>4000 bits</td>
<td>1200 bits</td>
<td>0.30</td>
<td>200 bits</td>
<td>200 bits</td>
<td>0.05</td>
<td>80 s</td>
</tr>
<tr>
<td>4000 bits</td>
<td>1600 bits</td>
<td>0.40</td>
<td>600 bits</td>
<td>600 bits</td>
<td>0.15</td>
<td>80 s</td>
</tr>
<tr>
<td>4000 bits</td>
<td>2000 bits</td>
<td>0.50</td>
<td>1000 bits</td>
<td>1000 bits</td>
<td>0.25</td>
<td>80 s</td>
</tr>
<tr>
<td>4000 bits</td>
<td>4000 bits</td>
<td>1.00</td>
<td>3000 bits</td>
<td>3000 bits</td>
<td>0.75</td>
<td>80 s</td>
</tr>
</tbody>
</table>

41
6.4.2 Solving the modified small inverse problem

All experiments were performed on Linux with a 600-MHz Intel Pentium III processor. Experiments were carried out using the LLL implementation available in Victor Shoup’s NTL package \[7\]. The integer \( d \in [\frac{3}{4} N^\delta, N^\delta] \) was chosen at random. Results are in Table 6.4.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( d )</th>
<th>( \delta )</th>
<th>( m )</th>
<th>( t )</th>
<th>rank</th>
<th># of short enough vectors</th>
<th># of independent polynomials</th>
<th>running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000 bits</td>
<td>260 bits</td>
<td>0.26</td>
<td>3</td>
<td>1</td>
<td>30</td>
<td>21</td>
<td>0</td>
<td>70 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>300 bits</td>
<td>0.30</td>
<td>3</td>
<td>1</td>
<td>30</td>
<td>17</td>
<td>0</td>
<td>70 s</td>
</tr>
<tr>
<td>1000 bits</td>
<td>320 bits</td>
<td>0.32</td>
<td>3</td>
<td>1</td>
<td>30</td>
<td>13</td>
<td>0</td>
<td>70 s</td>
</tr>
</tbody>
</table>

All experiments did not produce algebraically independent polynomials. Hence we were not able to recover \((x_0, y_0, z_0)\). All reduced polynomials had a common factor \( p(x, y, z) \).

6.4.3 Factoring with a hint

All experiments were performed under Linux on a 600-MHz Intel Pentium III processor. Experiments were carried out using the LLL implementation available in Victor Shoup’s NTL package \[7\]. All results ended in failure. Remember we already have one bivariate polynomial

\[
b(x, y) = PQ + Qx + Py + xy - N = 0.
\]

All experiments performed found algebraically dependent polynomials, with \( b(x, y) \) as a common factor. However for \( \alpha = 1/6 \), were were able to factor \( r_2(x, y) \) such that

\[
r_2(x, y) = b(x, y)g(x, y)
\]

and \( g(x_0, y_0) = 0 \). Hence, we were able to recover \( x_0 \) and \( y_0 \). However for \( \alpha = 1/6 \) requires knowing \( \frac{1}{3} \log_2 N \) high order bits of \( p \).

6.5 Conclusion and Improvements

This chapter stresses the importance of finding the necessary and sufficient conditions for the LLL algorithm to produce algebraically independent polynomials.
All experiments in this chapter were based on solving small solutions that are already related. The lattice attack on the modified small inverse problem attempts to find algebraically independent polynomials with roots \((x_0, y_0, z_0)\). However, \((x_0, y_0, z_0)\) is already related by \(p(x, y, z)\). A more randomized lattice reduction algorithm might help solve this problem. The same approach might fix the factoring attack as well.

If the modified small inverse problem was solved, one could continue to improve the bounds by applying geometrically progressive matrices as in \([3]\). However a more effective improvement would be to repeat the process of finding a new equation. This would lead to more variables and a much larger lattice. However, repeating this procedure seems to converge towards \(\delta < 0.5\). The most interesting fact in this chapter is that the secret key \(d > N^{0.292}\) can be expressed as

\[
d = C_{2,1}c_1 - C_{1,1}c_2,
\]

where integers \(C_{2,1}, C_{1,1}\) are known and integers \(|c_1|, |c_2| < e^{\delta-0.25}\) are unknown.
References


