On Finding Small Solutions of Modular Multivariate Polynomial Equations

Charanjit S. Jutla

IBM T. J. Watson Research Center,
Yorktown Heights, NY 10598, USA

Abstract. Let $P(\mathbf{x}) \equiv 0 (mod\, N)$ be a modular multivariate polynomial equation, in m variables, and total degree k with a small root $\mathbf{x_0}$. We show that there is an algorithm which determines $c(\geq 1)$ integer polynomial equations (in m variables) of total degree polynomial in $cmk \log N$, in time polynomial in $cmk \log N$, such that each of the equations has $\mathbf{x_0}$ as a root. This algorithm is an extension of Coppersmith's algorithm [2], which guarantees only one polynomial equation. It remains an open problem to determine $\mathbf{x_0}$ from these linearly independent equations (which may not be algebraically independent) in polynomial time. The algorithm can be used to attack an RSA scheme with small exponent in which a message is padded with random bits in multiple locations. Given two encryptions of the same underlying message with multiple random paddings of total size about 1/9 of the length N (for exponent 3 RSA), the algorithm can be used to obtain the message.

1 Introduction

Let $P(x)$ be a degree k polynomial in x over the integers. It is well known that for the modular equation $P(x) = 0 (mod\, p)$, where p is a prime, all integer roots $(mod\, p)$ can be determined in polynomial time (Berlekamp's algorithm). All integer roots for the integer equation $P(x) = 0$ can also be determined in polynomial time ([11]). However, the modular equation modulo a large composite N is not known to be solvable in polynomial time (unless the factorization of N is known). This is the basis of various public-key cryptographic schemes, the most well known of which is RSA [12].

Coppersmith [2] recently showed that if the modular equation $P(x) = 0$ $(mod\, N)$, has a small root $x_0 < N^{1/k}$, then this root can be determined in time polynomial in $k \log N$. He showed that using the lattice basis reduction algorithm ([11]), one obtains another polynomial $Q(x)$ over the integers, of degree polynomial in $k \log N$ (and in polynomial running time), such that $Q(x_0) = 0$. Then $Q(x)$ can be solved for x_0 using previously known algorithms.

Coppersmith [2] also showed that his technique can be applied to multivariate polynomials (in variables $\mathbf{x}$). However, only one integer multivariate polynomial equation was obtained, which is insufficient to determine the small root $\mathbf{x_0}$. He mentioned solving for $\mathbf{x_0}$ as an open problem. The problem is further complicated by the fact that even if one obtains several integer multivariate polynomial

equation, they may not be sufficiently independent to eliminate variables so as to obtain a univariate polynomial equation.

We make progress in solving this open problem by showing that one indeed gets several (linearly independent) integer multivariate polynomial equations satisfied by the small root $\mathbf{x_0}$. Let $P(\mathbf{x})$ be an integer polynomial of total degree k, in m variables. Then, if $P(\mathbf{x_0}) = 0 (mod\, N)$, we obtain ck multivariate integer polynomial equations each satisfied by the same root $\mathbf{x_0}$. The total degree of each of these polynomials is polynomial in $cmk \log N$. The running time of the algorithm is also polynomial in $cmk \log N$. However, it remains an open problem whether these linearly independent equations can actually be solved for $\mathbf{x_0}$ in polynomial time, by e.g. eliminating variables (using resultants -cf. [8]).

For example, one may have obtained two equations $(x^2 + 1)f(x, y) = 0$, and $(y^2+1)f(x, y) = 0$. They are definitely linearly independent, but the only integer solutions are the ones which satisfy $f(x, y) = 0$.

For simplicity, we do the analysis for the univariate case, as the multivariate case is a simple generalization (see [2]). We show that for the univariate degree k polynomial equation $P(x) = 0\ (mod\ N)$, if there is a small integer root $|x_0| < N^{1/k}$, then one obtains ck linearly independent polynomial equations each satisfied by x_0 $(ck \geq 1)$.

The algorithm in [2] was obtained by doing lattice basis reduction ([11]) of the dual lattice formed from the coefficients of $P(x)$ and its powers. The analysis of the dual lattice (instead of the primal) makes it even more difficult to prove (or disprove) that the equations obtained are algebraically independent. In section 4 we give an algorithm which works by doing lattice basis reduction of the primal lattice. It is a form of generalized Diophantine approximation ([11]). The fact that Coppersmith's result [2] can be obtained alternatively by working in the primal lattice was observed independently by Howgrave-Graham [5]. Infact, he goes on to prove that the two alternate approaches are equivalent.

Finally, we mention that finding small roots of modular polynomial equations has great practical significance. Consider the public key encryption scheme RSA [12]. The public key in this scheme is a large composite number N, a product of two secret primes. Also part of the public key is a number called the exponent e. A plaintext x has the following encryption c

$$c = x^e\ (mod\, N)$$

Suppose the plaintext x is actually obtained from a message by padding it with some random bits. For example, in 1024 bit RSA (i.e. N is 1024 bits long), suppose the message m is only 1000 bits. Then the plaintext x could have been set to m appended by r, where r is a 24 bit random number (padding).

As an application of the univariate case, Coppersmith [2] showed that for RSA with small exponent (say 3), two encryptions of the same message with random paddings reveals the message, as long as the padding is less than 1/9 of the length of N.

He also showed, how multiple paddings are also prone to similar attacks, if one can solve the multivariate modular polynomial equations for small roots. In the previous example, a plaintext x with double padding could have been obtained from m by both prefixing and suffixing it with random paddings of 12 bits each. In section 6 we give details of how such an attack is actually launched.

Recently, Bleichenbacher [1] has shown that the KMOV public key cryptosystem (based on elliptic curves modulo large composites) can be attacked using the multivariate algorithm. Essentially, given the ciphertext, and 2/3 of the plaintext, the other 1/3 of the plaintext can be obtained. In section 7 we summarize both the KMOV scheme, and the attack on the KMOV scheme.

The rest of the paper is organized as follows. Section 2 introduces lattices and the LLL lattice basis reduction algorithm. Section 3 gives the details of the algorithm and the analysis for the univariate case, using the dual lattice. Section 4 introduces the Diophantine Approximation variant, which is based on the primal lattice. Section 5 states the multivariate case. In section 6 and 7 we give applications of the multivariate case to RSA and KMOV. In Section 8, we give empirical evidence as to how the equations obtained are very independent, and rather random in nature.

2 Reduced Basis for Lattices

Let n be a positive integer. A subset L of the n-dimensional real vector space $\mathcal{R}^n$ is called a *lattice* if there exists a basis $b_1, b_2, ...b_n$ of $\mathcal{R}^n$ such that

$$L = \sum_{i=1}^{n} \mathcal{Z} b_i = \left\{ \sum_{i=1}^{n} r_i b_i : r_i \in \mathcal{Z} (1 \leq i \leq n) \right\}$$

In this situation we say that $b_1, b_2, ...b_n$ form a *basis* for L, or that they *span* L. We recall the *Gram-Schmidt orthogonalization process*. The vectors b_i^* $(1 \leq i \leq n)$ and the real numbers μ_{ij} $(1 \leq j < i \leq n)$ are inductively defined by

$$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{ij} b_j^*$$
$$\mu_{ij} = (b_i, b_j^*)/(b_j^*, b_j^*),$$

where $(,)$ denotes the ordinary inner product on $\mathcal{R}^n$. The *determinant* $det(L)$ of the lattice L is defined by

$$det(L) = |det(b_1, b_2, ...b_n)| = |det(b_1^*, b_2^*, ...b_n^*)|$$

where b_i and b_i^* are written as column vectors.
The *dual lattice* of L is the lattice spanned by the rows of the matrix $(b_1, ..., b_n)^{-1}$.
A basis $b_1, b_2, ...b_n$ for a lattice L is called *reduced* (LLL) if

$$\mu_{ij} \leq 1/2 \; for \; 1 \leq j < i \leq n, and$$

$$|b_i^* + \mu_{i\,i-1} b_{i-1}^*|^2 \geq 3/4 |b_{i-1}^*|^2 \; for \; 1 < i \leq n.$$

Lemma 1 ([11]): Let $b_1, b_2, ...b_n$ be a reduced basis for a lattice L in $\mathcal{R}^n$, and let $b_1^*, b_2^*, ...b_n^*$ be the Gram-Schmidt orthogonal vectors (as defined above). Then

$$|b_j|^2 \leq 2^{i-1} \cdot |b_i^*|^2 \; for \; 1 \leq j \leq i \leq n$$

$$|b_j^*|^2 \leq 2^{i-j} |b_i^*|^2 \; for \; 1 \leq j \leq i \leq n$$

$$det(L) \leq \prod_{i=1}^{n} |b_i| \leq 2^{n(n-1)/4} \cdot det(L),$$

$$|b_1| \leq 2^{(n-1)/4} \cdot det(L)^{1/n}$$

Lemma 2 ([11]): Let $L \subset Z^n$ be a lattice with basis $b_1, b_2, ...b_n$, and let $B \in \mathcal{R}$, $B \geq 2$, be such that $|b_i|^2 \leq B$ for $1 \leq i \leq n$. Then there is an algorithm which obtains a reduced basis in $O(n^4 \log B)$ arithmetic operations on integers of binary length $O(n \log B)$.

Lemma 3 (Diophantine Approximation) ([11]): There exists a polynomial time algorithm that, given a positive integer n and rational numbers $\alpha_1, \alpha_2, ...\alpha_n, \epsilon$ satisfying $0 < \epsilon < 1$, finds integers $p_1, p_2, ...p_n, q$ for which

$$|p_i - q\alpha_i| \leq \epsilon \; for \; 1 \leq i \leq n,$$

$$1 \leq q \leq 2^{n(n+1)/4} \epsilon^{-n}$$

Lemma 3 follows from Lemma 2.

3 Obtaining more than one polynomial equations

We show that for a polynomial equation of degree $k (\text{mod } N)$ in one variable x, if there is a small root (smaller than $N^{1/k}$), then the same solution satisfies two (in-fact more than two) polynomial equations over integers of degree polynomial in k. Moreover, these polynomial equations can be obtained in time polynomial in $k \log N$.

The following is essentially an improved analysis of the technique outlined in [2], along with three new observations which we point out towards the end of this section.

Let x_0 be the small unknown solution of $P(x) = 0 \; (\text{mod } N)$. W.l.o.g assume that $P(x)$ is monic. Let $\epsilon = r/\log N$, where $r > 1$. Let $c \geq 1/k$ be a constant. The intention is to obtain ck different polynomial equations. Let h be an integer such that $hk > \max(7, (4c \log N)/(r-1))$.
Let $n = hk$.

For each pair of integers i, j satisfying $0 \leq i < k$, $1 \leq j < h$, set

$$Q_{ij}(x) = x^i P(x)^j$$

Thus, $Q_{ij}(x_0) = x_0^i y_0^j N^j$, where $y_0 = P(x_0)/N$. ...(1)

Build the rational matrix M of size $(2hk-k) \times (2hk-k)$ as in section 2 of [2]. Let,

$$M = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$$

The upper right block B, of size $(hk) \times (hk-k)$, has rows indexed by the integer g with $0 \leq g < hk$, and columns indexed by $\gamma(i,j) = i + (j-1)k$, with $0 \leq i < k$ and $1 \leq j < h$, so that $0 \leq \gamma(i,j) < hk - k$. The entry $B[g, \gamma(i,j)]$ is the coefficient of x^g in the polynomial $Q_{ij}(x)$.

The lower right block C is a $(hk-k) \times (hk-k)$ diagonal matrix, with the value N^j in each column $\gamma(i,j)$. The upper left block A is a $(hk) \times (hk)$ diagonal matrix. $A[g,g]$ has the rational approximation to $\delta * X^g$, where $X = N^{(1/k)-\epsilon}$, and $\delta = 1/\sqrt{(hk)}$.

We restrict our attention to $|x_0| < X$. ...(2)

Let row vector r be such that its left-hand entries are $r_g = x_0^g$, and whose right-hand entries are $r_{\gamma(i,j)} = -x_0^i y_0^j$. Let $s = rM$. From (1) and (2), it follows that the Euclidean norm of s is strictly less than 1. (see [2]).

The matrix M can also be written as

$$M = \begin{pmatrix} A1 & B1 \\ A2 & B2 \\ 0 & C \end{pmatrix}$$

where $B2$ is an upper triangular matrix$((hk-k) \times (hk-k))$, and the diagonal entries of $B2$ are all one (since all the polynomials Q_{ij} are monic). Thus, we can transform M, using elementary row operations, to M'

$$M' = \begin{pmatrix} A1' & 0 \\ A2' & I \\ A3' & 0 \end{pmatrix}$$

where I is an identity matrix $((hk-k) \times (hk-k))$. Note that $(A1'^{\top}\ A2'^{\top})^{\top}$ is still upper triangular. It also follows that $(A1'^{\top}\ A3'^{\top})^{\top}$ is upper triangular. In fact, we will soon calculate the diagonal entries of $(A1'^{\top}\ A3'^{\top})^{\top}$.

Obtain $\tilde{M}$ from M' using row exchanges.

$$\tilde{M} = \begin{pmatrix} A1' & 0 \\ A3' & 0 \\ A2' & I \end{pmatrix}$$

Denote the upper left $(hk) \times (hk)$ block of $\tilde{M}$ by $\hat{M} = (A1'^{\top}\ A3'^{\top})^{\top}$. Note that

$$det(\hat{M}) = det(M)$$

Now we restrict our attention to $\hat{M}$. By construction of M, and the row operations performed, $\hat{M}$ is still an upper triangular matrix. The lower diagonal

elements of $\hat{M}$, for $k \leq \gamma(i,j) + k < hk$, are given by $\hat{M}_{\gamma(i,j)+k,\gamma(i,j)+k} = -N^j \delta X^{-(\gamma(i,j)+k)}$.

The next step in [2] is to do the LLL basis reduction of the lattice generated by rows of $\hat{M}$ (see Lemma 2). The LLL algorithm goes through a series of steps involving elementary operations on the rows of the matrix representing the lattice. Let, b_i, and b_i^* denote the intermediate basis vectors and the corresponding Gram-Schmidt orthogonalization vectors respectively. It is a property of the LLL algorithm that $\max \{|b_i^*|^2 : 1 \leq i \leq n\}$ is non-increasing.

Since $\hat{M}$ is an upper triangular matrix, if we start the LLL algorithm on $\hat{M}$, with the order of the rows reversed, the Gram-Schmidt orthogonalization of this initial matrix will yield a diagonal matrix, with the diagonal entries remaining the same. The maximum of the absolute value of these diagonal entries is $N^{h-1}\delta X^{-(k+(h-2)k)}$. Denote this quantity by $bmax$.

A property of the reduced basis is that the final b^* satisfy (see Lemma 1)

$$|b_j^*|^2 \leq 2^{i-j}|b_i^*|^2 \text{ for } 1 \leq j \leq i \leq n$$

Thus,

$$\prod_{j=1}^{i} |b_j^*|^2 \leq 2^{i(i-1)/2}|b_i^*|^{2i}$$

Moreover,

$$\prod_{j=1}^{n} |b_j^*| = det(\hat{M})$$

Thus, $|b_i^*| \geq (det(\hat{M})/bmax^{(n-i)})^{1/i} 2^{-(i-1)/4}$.

With the choice of h as previously mentioned, it can be shown that (recall, $det(\hat{M}) = det(M) = N^{(h-1)hk/2} X^{-hk(hk-1)/2} \delta^{hk}$. Also, for $hk > 7$, $\delta^{hk} > 2^{(hk-1)/2}$)

$$det(\hat{M})/bmax^{ck} > 2^{n(n-1)/4}$$

Thus, for $n - ck \leq i \leq n$, $|b_i^*| > 1$.

The Euclidean norm of any element $\sum c_i b_i$ of the lattice $\hat{M}$ is at least $|c_n| \times |b_n^*| > |c_n|$. So any lattice element with norm less than 1 must have $c_n = 0$. Continuing this argument, we have that any such lattice element must have $c_i = 0$, for $n - ck \leq i \leq n$. In particular s is such an element.

Let $s = rM = rI'\hat{M} = rI''\bar{M}$, where $\bar{M}$ is the matrix obtained by performing the LLL algorithm above. Now, rI'' has the property that the entries with index i, $n - ck \leq i \leq n$, are zero. Thus, the same holds for $s\bar{M}^{-1}$. Since, s has only powers of x as variables, this yields ck equations in $x, ..x^{n-1}$. These equations are linearly independent as $\bar{M}$ is non-singular.

These polynomial equations over integers can be solved to obtain x_0.

Summarizing, the improved analysis was based on the following observations:

1. One can do elementary row operations on M so that $\hat{M}$ is upper triangular, and can give the diagonal entries of $\hat{M}$ explicitly.
2. These diagonal entries give initial values of $|b_i^*|$.
3. The maximum norm $|b_i^*|$ is non-increasing as LLL proceeds.

4 Diophantine Approximation Alternative

The analysis in the previous section (as in [2]) was done with the dual lattice formed from the equations $Q_{ij}(x)$, i.e. the lattice was reduced with respect to the rows instead of the columns. Further, at the end we had to invert the matrix $\bar{M}$ to obtain the polynomial equations over the integers. Thus, it is difficult to get an insight into how the integer equations were obtained from the original polynomials $Q_{ij}(x)$. Without such an insight it would be even more difficult to prove if the equations obtained are algebraically independent.

In this section, we do a similar analysis but with the primal lattice, and show that the integer polynomial equations are obtained by a weighted (and generalized) Diophantine approximation (see Lemma 3).

Let $P(x) = a_0 + a_1x + ... + a_kx^k$. Since x_0 is a root of this polynomial modulo N, there is an integer y_0: $a_0/N + a_1/Nx_0 + ... + a_k/Nx_0^k = y_0$. If each of the coefficients was small (e.g. ≤ 1), and x_0 was small (which we already know is small), y_0 would be zero, and we would have an equation over the integers. However, a_i in general is not small.

Let $|x_0| < X$. Consider the equation,

$$(p_0 + qa_0/N) + (p_1 + qa_1/N)x_0 + ...(p_k + qa_k/N)x_0^k = qy_0 + \sum_{i=0}^{k} p_i x_0^i$$

Here p_i and q are integers. If we can ensure that each of the quantities $|(p_i + qa_i/N)X^i|$ is less than $1/(k+1)$, then since the right hand side of the equation is an integer it will have to be zero. We would have an integer polynomial equation as long as one of the coefficients is non-zero. Since $a_k = 1$ (P is monic), the latter condition is guaranteed by requiring $0 < |q| < N$

The former condition can be fulfilled by doing a simultaneous Diophantine approximation of a_i/N (weighted by X^i) – see section 2. However, the requirement $|q| < N$ does not allow for a good enough approximation, and hence requires X to be rather small ($X < N^{1/k^2}$).

We now employ the trick used in [2]; we use all of the polynomials Q_{ij} (as defined in Section 3). Let h be an integer to be determined. Let $\gamma(i,j)$ be as defined in the previous section. Consider the following equation which holds for x_0, $y_0 = P(x_0)/N$, all integers p_g and $q_{\gamma(i,j)}$:

$$\sum_{g=0}^{hk-1} (p_g + \sum_{i=0}^{k-1}\sum_{j=1}^{h-1} q_{\gamma(i,j)}a_{g,\gamma(i,j)}/N^j)x_0^g = \sum_{g=0}^{hk-1} p_g x_0^g + \sum_{i=0}^{k-1}\sum_{j=1}^{h-1} q_{\gamma(i,j)}x_0^i y_0^j \quad (3)$$

Here, $a_{g,\gamma(i,j)}$ is the coefficient of x^g in the polynomial $Q_{ij}(x)$.

If we can upper bound each of the outer summands on the LHS (in absolute value) by $1/hk$, then since the right hand side is an integer, it must be zero. This would yield an integer polynomial equation. It would be non-trivial if each $|q_{\gamma(i,j)}| < N^j$, and some $|q_{\gamma(i,j)}| > 0$.

To this end, we set up a matrix M as in the previous section, again intending to do a (generalized) weighted Diophantine approximation. Let X (to be determined) be an upper bound on x_0.

$$M = \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}$$

The upper right block B, of size $(hk) \times (hk - k)$, has entry $B[g, \gamma(i,j)] = (a_{g,\gamma(i,j)}/N^j)X^g$.
The lower right block C is a $(hk - k) \times (hk - k)$ diagonal matrix, with the value μ_j (to be determined) in each column $\gamma(i,j)$. The upper left block A is a $(hk) \times (hk)$ diagonal matrix with $A[g,g] = X^g$.

We perform lattice basis reduction on the lattice generated by the columns of M. Let b_1 be the first vector in the reduced basis, which is a linear combination of the columns of M, given by integers p_g and $q_{\gamma(i,j)}$. Thus, each of the outer summands on the LHS of equation (3) is less than $|b_1|$.

It is a property of the reduced basis that $|b_1| < 2^{(n-1)/4}(det(M))^{1/n}$, where $n = 2hk - k$ is the dimension of M. Thus, we require

$$2^{(n-1)/4}(det(M))^{1/n} < 1/hk \qquad ...(4)$$

Also, $|q_{\gamma(i,j)} * \mu_j| < |b_1|$. Thus, to ensure $|q_{\gamma(i,j)}| < N^j$, it suffices to have

$$|b_1| < 2^{(n-1)/4}(det(M))^{1/n} < |\mu_j|N^j \qquad ...(5)$$

Some $q_{\gamma(i,j)}$ will be non-zero if $|b_1| < 1$.

Also, note that $det(M) = X^{hk(hk-1)/2}\mu_1^k ... \mu_{h-1}^k$. ...(6)
The inequalities (4) and (5) impose opposing conditions on μ_j. A simple calculation shows that both (4) and (5) can be satisfied if $X < N^{1/k-\epsilon}$, where $\epsilon > 1/\log N$. This requires h to be $O(\log N)$ as well.

To obtain more than one equation, we bound other small vectors b_2, b_3 etc. of the reduced basis. Thus, if $|b_t| < 1/hk$, b_t would yield another set of integers p_g, and $q_{\gamma(i,j)}$, and hence another polynomial equation, by (3).

First note that, since M is an upper-triangular matrix, and b_t is an integer linear-combination of the columns of M, therefore $|b_t| > \min(|M_{ii}|)$, $0 < i < n$. Call this quantity M_{min}. Another property of the reduced basis is that $|b_j| < 2^{(i-1)/2}|b_i^*|$, for $1 \leq j \leq i \leq n$. It follows that,

$$|b_j| \leq 2^{n(n-1)/(4(n-(j-1)))}(det\ M)^{1/(n-(j-1))}M_{min}^{-(j-1)/(n-(j-1))}, for\ 1 \leq j \leq n$$

However, $M_{min} = \min(|M_{ii}|) = \min(X^g, \mu_j)$, where $0 \leq g < hk - 1$, and $1 \leq j < h$. The previous analysis had forced $\mu_j \approx N^{-j}$. Hence $M_{min} \approx N^{-(h-1)}$. This does not yield a good upper bound for $|b_j|$.

To solve this problem, we do certain elementary column operations (similar to the row operations of the previous section), ending in yet another diagonal $(n \times n)$ matrix. View M as

$$M = \begin{pmatrix} A1 & B1 \\ A2 & B2 \\ 0 & C \end{pmatrix}$$

We make use of the fact that the polynomials $q_{ij}(x)$ are monic, and hence, we can obtain the following matrix M'

$$M' = \begin{pmatrix} A1' & B1' \\ 0 & B2' \\ A3' & C' \end{pmatrix}$$

where $B2'$ is now a diagonal matrix, and $(A1'^{\mathsf{T}}\ A3'^{\mathsf{T}})^{\mathsf{T}}$ is an upper triangular matrix. Thus, we can perform LLL on the following upper-triangular matrix instead

$$\tilde{M} = \begin{pmatrix} A1' & B1' \\ A3' & C \\ 0 & B2' \end{pmatrix}$$

The minimum diagonal entry of this matrix is $\mu_{h-1} N^{h-1}$. The determinant of $\tilde{M}$ is same as that of M, and the previous analysis holds. This allows us to obtain bounds for X and the corresponding h (for a given c) similar to those in Section 3.

5 Multivariate polynomials

Let $P(\mathrm{x}) \equiv 0\ (mod\ N)$ be a modular multivariate polynomial equation, in m variables, and total degree k with a root $\mathrm{x_0}$. Moreover, let the root be small, i.e. $|x_{0i}| < N^{\alpha_i}$, for $1 \leq i \leq m$, and $\sum \alpha_i < (1/k)$. There is an algorithm which determines $ck (ck \geq 1)$ linearly independent integer polynomial equations (in m variables) of total degree polynomial in $cmk \log N$, in time polynomial in $cmk \log N$, such that each of the equations has $\mathrm{x_0}$ as a root.

The proof of this statement is exactly the same as that of the univariate case (Section 3 or 4), except that the polynomials q_{ij} and the matrix M are built in a more generalized fashion (see [2]). For completeness, we reproduce from [2].

Define

$$z = P(x_{01}, \ldots x_{0m})/N$$

$$q_{i_1 \ldots i_m j}(x_1, \ldots, x_m) = x_1^{i_1} \cdots x_m^{i_m} P(x_1, \ldots x_m)^j$$

Note that $q_{i_1 \ldots i_m j}(x_{01}, \ldots, x_{0m})$ is divisible by N^j. Set a limit T (corresponding to hk in section 3) and develop the modular equations $q_{i_1 \ldots i_m j}(x_{01}, \ldots, x_{0m}) = 0\ (mod\ N)$ for all nonnegative integer indices $(i_1, \ldots i_m, j)$ with $1 \leq j$, $i_m < k$, and $i_1 + i_2 + \ldots + i_m + kj \leq T$. The condition $i_m < k$ is required so that the

submatrix $B2$ (as in section 3) is a uppertriangular matrix with diagonal entries all one (w.l.o.g. we can assume that the polynomials are monic in x_m).

We build the matrix M analogous to that of section 3. The vector r contains all monomials of total degree at most T, so the sum of the total degrees of these monomials is (see e.g. [7])

$$\sum_{i=0}^{T} i \binom{i+m-1}{m-1} = \sum_{i=0}^{T} m \binom{i+m-1}{m} = m \binom{T+m}{m+1}$$

By symmetry, the sum of the degrees in each x_i, $i = 1, ..., m-1$, is $\binom{T+m}{m+1}$ These appear as negative exponent of α_i in the diagonal entries of the upper left block A of M.

Let $T = hk$. The powers (moduli) of N appearing in the lower right submatrix C of M add (asymptotically for large h) to

$$\sum_{j=1}^{h} \binom{hk-jk+m}{m} - \sum_{j=1}^{h} \binom{hk-(j+1)k+m}{m} = \frac{1}{k}\binom{T+m}{m+1} - O(\binom{T+m}{m})$$

For example, for $m = 2$, the powers of N add upto $k^2(h-1)h(2h-1)/12$. With the requirement $\Sigma\alpha_i < (1/k) - \epsilon$ for appropriate ϵ we will have $|b_i^*| > 1$, for $n - ck \leq i \leq n$, where n is the dimension of the matrix $\hat{M}$. Note that the dimension of the matrix $\hat{M}$ is the total number of monomials which is $\binom{T+m}{m}$.

6 Application to RSA with Random Paddings

For simplicity, let's assume that the public exponent in the RSA scheme is 3. Given a plaintext x, the ciphertext c obtained by encrypting x under RSA is given by

$$c = x^3 \ (mod\ N)$$

Here N is a large composite, which is part of the public key.

If a message m is not of full length (for example, if m is less than 1024 bits, when N is 1024 bit long), one could pad m with a random number t to obtain a number x of full size. Thus, if m was only 1000 bits long, a 24 bit random string t could be appended to m to obtain x, in which case

$$x = 2^{24} \cdot m + t$$

Alternatively, more than one random number could have been used to fill different contiguous locations of x. For example, the top 12 bit positions, and the least significant 12 bits could be used for padding with $t1$ and $t2$ respectively, whereas m could be placed in the middle, in which case

$$x = 2^{1012} \cdot t1 + 2^{12} \cdot m + t2$$

If a message is encrypted twice using these schemes, then the results of the previous sections can be used (heuristically) to obtain the message as follows. As an example, we will consider the case of double padding mentioned above.

Note that, is we have two encryptions of m, i.e. of plaintexts x and x', both of which were obtained from m by double random paddings then

$$x' = x + 2^{1012} \cdot (t1' - t1) + (t2' - t2)$$

Let $t = 2^{1012} \cdot (t1' - t1) + (t2' - t2)$. Then

$$c = x^3 \,(mod\, N)$$

$$c' = x'^3 \;=\; (x+t)^3 \;=\; x^3 + 3x^2t + 3xt^2 + t^3 (mod\, N)$$

Using a result of Franklin and Reiter [4] (also see [3]) it follows that

$$x = \frac{t(c' + 2c - t^3)}{c' - c + 2t^3} \;=\; \frac{t(3x^3 + 3x^2t + 3xt^2)}{3x^2t + 3xt^2 + 3t^3} \,(mod\, N)$$

Hence, we can recover x if we know c, c', N, and t.

If t is not known, but the random paddings were small, i.e., each of $(t1' - t1)$, and $(t2' - t2)$ were small (say, less than $N^{1/18}$), then we could do the following. First, we eliminate x, by taking the resultant ([8]) of the above two equations with respect to x. Thus,

$\text{Resultant}_x(x^3 - c, (x+t)^3 - c') =$,

$t^9 + (3c - 3c')t^6 + (3c^2 + 21cc' + 3(c')^2)t^3 + (c - c')^3 \;= 0 \,(mod\, N)$

This is a multivariate polynomial in $(t1' - t1)$, $(t2' - t2)$, with small roots. Thus, we can apply the algorithm of section 5, and there is a possibility that it may yield $(t1' - t1)$, and $(t2' - t2)$, and hence t.

In section 8 we show some empirical results, and mention the practicality of this attack.

7 Application to Attacks on KMOV

Let N be a large composite integer, product of two secret prime numbers p and q. The public key of the KMOV [6] public-key scheme is N and an integer e relatively prime to $(p+1)(q+1)$. A message is a pair (m_x, m_y) where $m_x, m_y \in \mathcal{Z}/(N)$. It is encrypted by computing $(c_x, c_y) \;=\; e \cdot (m_x, m_y) \,(mod\, N)$, over an elliptic curve parameterized by $b = m_y^2 - m_x^3 \,(mod\, N)$. The scalar multiplication by e refers to adding (m_x, m_y) to itself e times in the group of the elliptic curve (see [9],[6] or [1] for definition of an elliptic curve). We will not go into the details of this group, but it suffices to note that a point $(x1, y1)$ is on the elliptic curve parameterized by b if $y1^2 = x1^3 + b \,(mod\, N)$.

Since, both the plaintext (m_x, m_y) , and the ciphertext (c_x, c_y) are on this elliptic curve, it follows that

$$b = c_y^2 - c_x^3 \;= m_y^2 - m_x^3 \,(mod\, N)$$

Bleichenbacher ([1]) observed that if the ciphertext is known, which implies that b is known, and 2/3 of the plaintext (m_x, m_y) is known, then the rest of the plaintext can be obtained (using the results from section 5) from the degree 3, multivariate polynomial equation obtained from

$$m_x^3 = m_y^2 + b \, (mod \, N)$$

8 Empirical results

We implemented the algorithm for the univariate case using the package LiDIA [10]. Surprisingly, not only did we we obtain a large number of equations, but the GCD of the polynomials turned out to be a polynomial of a very small degree. For example, the following polynomial $P(x)$, has root $x_0 = 1$ (modulo $N = 84$).

$$P(x) = 83 + 42x + 42x^2 + x^3$$

The algorithm of Section 3, yielded the following polynomials (for $h = 3$), each of which evaluated to zero at $x_0 = 1$

$$\begin{array}{cccccccc}
-1 & +x & -x^2 & +2x^3 & -2x^4 & +2x^5 & -x^6 & +x^7 & -x^8 \\
-1 & & & +2x^3 & & & -x^6 & & \\
-1 & -3x & -x^2 & +2x^3 & +6x^4 & +2x^5 & -x^6 & -3x^7 & -x^8 \\
-37 & & & +32x^3 & & & +5x^6 & & \\
 & 37x & & & -32x^4 & & & -5x^7 & \\
-1 & +x & +73x^2 & +2x^3 & -2x^4 & -62x^5 & -x^6 & +x^7 & -11x^8 \\
1607 & -1607x & & +146x^3 & -146x^4 & & +11x^6 & -11x^7 & \\
1610 & +x & -1608x^2 & +140x^3 & -2x^4 & -144x^5 & +14x^6 & +x^7 & -12x^8
\end{array}$$

The GCD of these polynomials is $(x - 1)$.

When dealing with multivariate polynomials, the dimension of the matrix involved in the basis reduction starts becoming rather large. As mentioned at the end of section 5 the size of such a matrix for bivariate polynomials (i.e $m = 2$) is square that of $T = hk$. If we want to detect upto $1/k$ of the missing bits, then the running time of the algorithm is proportional to $2^{\log N/(hk)}$, since $\epsilon = r/\log N$ (see the beginning of section 3). Since, the LLL algorithm itself has time complexity $O(n^6 \log^3 N^h)$, where $n = (hk)^2$ is the dimension of the basis, the total time complexity is $O(2^{\log N/(hk)} h^{15} k^{12} \log^3 N)$. For 1024 bit RSA, $k = 3$, h has to be rather large, which makes the algorithm rather impractical for current computers. If we are only interested in detecting much fewer missing bits, then the size of the matrix and hence the running time may be more amenable, and actually feasible.

9 Conclusion

We make progress in solving modular multivariate polynomial equations with small roots. Our algorithm obtains several integer polynomial equations in the

multiple variables. However, the problem of showing that these equations can actually be used to determine the roots is still open. The problem lies in the fact that even though the equations may be linearly independent, one may not be able to eliminate variables by taking resultants.

However, empirical results show that the different equations obtained are rather random in nature, and hence their resultant may be able to eliminate all but one variable. On the other hand, even though the algorithm is polynomial in k, and $\log N$, the exponents are rather large for the multivariate case. Thus, the algorithm is not practical (for current computers) if the size of the root is as large as $N^{1/k}$. Thus for instance, for 1024 bit RSA with exponent 3, it may not be practical to launch an attack to detect 340 missing bits in the plaintext (spread over two contiguous regions). However, if much lesser number of bits are missing, the algorithm can be used effectively to detect those bits.

Acknowledgments

The author would like to thank Don Coppersmith for helpful comments.

References

1. Daniel Bleichenbacher, On the Security of the KMOV Public Key Cryptosystem, in *Proc. Crypto 97*, LNCS 1294, pp 235-248.
2. Don Coppersmith, Finding a Small Root of a Univariate Modular Equation, in *Eurocrypt 96*, LNCS 1070, pp 155-165.
3. D. Coppersmith, M. Franklin, J. Patarin and M. Reiter, *Low exponent RSA with related messages*, Proc. Eurocrypt 96.
4. M . Franklin,and M. Reiter, *A linear protocol failure for RSA with exponent three*, presented at CRYPTO95 rump session.
5. N. Howgrave-Graham, *Finding small roots of univariate modular equations revisited*, Cryptography and Coding, LNCS 1355, M. Darnell, Ed. Springer-Verlag, 1997, pages 131-142
6. K. Koyama, U. Maurer, T. Okamata, and S. Vanstone, New public-key schemes based on elliptic curves over the ring Z_n. In Advances in Cryptology - CRYPTO 91, LNCS vol. 576, pp 252-266.
7. D.E. Knuth, *The art of computer programming*, Vol 1, Fundamental algorithms, Addison Wesley 1973
8. D.E. Knuth, *The art of computer programming*, Vol 2, Seminumerical algorithms, Addison Wesley 1981
9. N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of Computation, 48(177):203-209,1987.
10. LiDIA,
A C++ Library For Computational Number Theory, http://www.informatik.th-darmstadt.de/TI/LiDIA/
11. A.K. Lenstra, H.W. Lenstra, and L.Lovász, Factoring Polynomials with Rational Coefficients, in *Mathematische Annalen*, Vol. 261, pp 515-534, 1982
12. R.L. Rivest, A. Shamir and L. Adleman, *A method of obtaining digital signatures and public-key cryptosystems*, CACM, Vol 21, no. 2, Feb 1978.