APPROXIMATE COMMON DIVISORS VIA LATTICES

HENRY COHN AND NADIA HENINGER

Abstract. We analyze the multivariate generalization of Howgrave-Graham's algorithm for the approximate common divisor problem. In the m-variable case with modulus N and approximate common divisor of size N^β, this improves the size of the error tolerated from N^{β^2} to $N^{\beta^{(m+1)/m}}$, under a commonly used heuristic assumption. This gives a more detailed analysis of the hardness assumption underlying the recent fully homomorphic cryptosystem of van Dijk, Gentry, Halevi, and Vaikuntanathan. While these results do not challenge the suggested parameters, a $2^{\sqrt{n}}$ approximation algorithm for lattice basis reduction in n dimensions could be used to break these parameters. We have implemented our algorithm, and it performs better in practice than the theoretical analysis suggests.

Our results fit into a broader context of analogies between cryptanalysis and coding theory. The multivariate approximate common divisor problem is the number-theoretic analogue of noisy multivariate polynomial interpolation, and we develop a corresponding lattice-based algorithm for the latter problem. In particular, it specializes to a lattice-based list decoding algorithm for Parvaresh-Vardy and Guruswami-Rudra codes, which are multivariate extensions of Reed-Solomon codes. This yields a new proof of the list decoding radii for these codes.

1. Introduction

Given two integers, we can compute their greatest common divisor efficiently using Euclid's algorithm. Howgrave-Graham [15] formulated and gave an algorithm to solve an approximate version of this question, asking the question "What if instead of exact multiples of some common divisor, we only know an approximation?" In the simplest case, we are given one exact multiple $N = pq_0$ and one near multiple $a_1 = pq_1 + r_1$, and the goal is to learn p, or at least $p \gcd(q_0, q_1)$.

In this paper, we generalize Howgrave-Graham's approach to the case when one is given many near multiples of p. The hardness of solving this problem for small p (relative to the size of the near multiples) was recently proposed as the foundation for a fully homomorphic cryptosystem [11]. Specifically, we can show that improving the approximation of lattice basis reduction for the particular lattices L we are looking at to $2^{\sqrt{\dim L}}$ would break the suggested parameters in the system. See Section 3 for the details. The approximate common divisor problem is also closely related to the problem of finding small solutions to multivariate polynomials, a problem first posed by Coppersmith [7], and whose various extensions have many applications in cryptanalysis [3].

Date: August 12, 2011.

Key words and phrases. Coppersmith's algorithm, lattice basis reduction, approximate common divisors, fully homomorphic encryption, list decoding, Parvaresh-Vardy codes, noisy polynomial interpolation.

The multivariate version of the problem allows us to improve the bounds for when the approximate common divisor problem is solvable: given $N = pq_0$ and m approximate multiples $a_i = pq_i + r_i$ of $p = N^\beta$, as well as upper bounds X_i for each $|r_i|$, we can find the perturbations r_i when

$$\sqrt[m]{X_1 \dots X_m} < N^{(1+o(1))\beta^{(m+1)/m}}.$$

In other words, we can compute approximate common divisors when r_i is as large as $N^{\beta^{(m+1)/m}}$. For $m = 1$, we recover Howgrave-Graham's theorem [15], which handles errors as large as N^{β^2}. As the number m of samples grows large, our bound approaches N^β, i.e., the size of the approximate common divisor p. Our algorithm runs in polynomial time for fixed m. We cannot rigorously prove that it always works, but it is supported by a heuristic argument and works in practice.

There is an analogy between the ring of integers and the ring of polynomials over a field. Under this analogy, finding a large approximate common divisor of two integers is analogous to noisy polynomial interpolation, as we explain in Section 1.2.2. One of the most important applications of noisy polynomial interpolation is decoding of Reed-Solomon codes. Guruswami and Sudan [13] increased the feasible decoding radius of these codes by giving a list-decoding algorithm that outputs a list of polynomially many solutions to a noisy polynomial interpolation problem. The analogy between the integers and polynomials was used in [6] to give a proof of the Guruswami-Sudan algorithm inspired by Howgrave-Graham's approach.

Parvaresh and Vardy [21] developed a related family of codes with a larger list-decoding radius than Reed-Solomon codes. The decoding algorithm corresponds to noisy simultaneous interpolation of several polynomials.

In this paper, we observe that the problem of noisy simultaneous interpolation of multiple polynomials is the exact analogue of the approximate common divisor problem with many inputs, and the improved list-decoding radius of Parvaresh-Vardy codes corresponds to the improved error tolerance in the integer case. We adapt our algorithm for the integers to give a corresponding algorithm to solve the noisy multiple polynomial interpolation problem.

1.1. **Related work.** Howgrave-Graham first posed the problem of approximate integer common divisors in [15], and used it to address the problem of factoring when information is known about one of the factors. His algorithm gave a different viewpoint on Coppersmith's proof [7] that one can factor an RSA modulus $N = pq$ where $p \approx q \approx \sqrt{N}$ given the most significant half of the bits of one of the factors. This technique was applied by Boneh, Durfee, and Howgrave-Graham [4] to factor numbers of the form $p^r q$ with r large. Jutla [16] considered the problem of finding small solutions to multivariate polynomial equations modulo integers, and showed how to do so by obtaining several equations satisfied by the desired roots using lattice basis reduction. Herrmann and May [14] gave a similar algorithm in the case of finding solutions to multivariate linear equations modulo divisors of a given integer. They apply their results to the case of factoring with bits known when those bits might be spread across $\log \log N$ chunks of p. Notably, their results display similar behavior to ours as the number of variables grows large. Van Dijk, Gentry, Halevi, and Vaikuntanathan [11] discuss extensions of Howgrave-Graham's method to larger m and provide a rough heuristic analysis in Appendix B.2 of the longer version of their paper available on the Cryptology ePrint Archive.

In addition to the extensive work on noisy polynomial interpolation in the coding theory literature, the problem in both the single and multiple polynomial cases has been used as a cryptographic primitive, for example in [19], [18], and [1] (broken in [9]). Coppersmith and Sudan [8] gave an algorithm for simultaneous interpolation of multiple polynomials, assuming random (rather than adversarially chosen) errors. Bleichenbacher, Kiayias, and Yung [2] gave a different algorithm for simultaneous interpolation of multiple polynomials under a similar probabilistic model. Parvaresh and Vardy [21] were the first to beat the list-decoding performance of Reed-Solomon codes for adversarial errors, by combining multiple polynomial interpolation with carefully chosen constraints on the polynomial solutions; this allowed them to prove that their algorithm ran in polynomial time, without requiring any heuristic assumptions. Finally, Guruswami and Rudra [12] combined the idea of multi-polynomial reconstruction with an optimal choice of polynomials to construct codes that can be list-decoded up to the information-theoretic bound (for large alphabets).

1.2. **Problems and results.**

1.2.1. *Approximate common divisors.* Following Howgrave-Graham, we define the "partial" approximate common divisor problem to be the case when one has $N = pq_0$ and m approximate multiples $a_i = pq_i + r_i$ of p. We want to recover an approximate common divisor. To do so, we will compute $r_1, \ldots, r_m$, after which we can simply compute the exact greatest common divisor of $N, a_1 - r_1, \ldots, a_m - r_m$.

If the perturbations r_i are allowed to be as large as p, then it is clearly impossible to reconstruct p from this data. If they are sufficiently small, then one can easily find them by a brute force search. The following heuristic theorem interpolates between these extremes: as m grows, the bound on the size of r_i approaches the trivial upper bound of p.

Heuristic Theorem 1 (Partial approximate common divisors)**.** *Given generic positive integers $N, a_1, \ldots, a_m$ and bounds $\beta \gg 1/\sqrt{\log N}$ and $X_1, \ldots, X_m$, we can find all $r_1, \ldots, r_m$ such that*

$$\gcd(N, a_1 - r_1, \ldots, a_m - r_m) \geq N^\beta$$

and $|r_i| \leq X_i$, provided that

$$\sqrt[m]{X_1 \ldots X_m} < N^{(1+o(1))\beta^{(m+1)/m}}.$$

The algorithm runs in polynomial time for fixed m, and the $\gg$ and $o(1)$ are as $N \to \infty$.

For $m = 1$, this theorem is rigorous and due to Howgrave-Graham [15]. For $m > 1$, by "generic" integers we mean integers chosen at random from a large range, or at least giving independent information in some sense. Specifically, we must rule out attempting to improve on the $m = 1$ case by deriving $a_2, \ldots, a_m$ from a_1, for example by taking a_i to be a small multiple of a_1 plus an additional perturbation (or, worse yet, $a_1 = \cdots = a_m$).

We describe our algorithm to solve this problem in Section 2. We follow the general technique of Howgrave-Graham: we use LLL lattice basis reduction to construct m polynomials for which $r_1, \ldots, r_m$ are roots, and then we solve the system of equations.

This algorithm is heuristic, because we assume we can obtain m short lattice vectors representing algebraically independent polynomials from the lattice that

we will construct. This assumption is commonly made when applying multivariate versions of Coppersmith's method, and has generally been observed to hold in practice. See Section 2 for more details. This is where the restriction to generic inputs becomes necessary: if $a_1, \dots, a_m$ are related in trivial ways, then the algorithm will simply recover the corresponding relations between $r_1, \dots, r_m$, without providing enough information to solve for them.

Note that we are always able to find one nontrivial algebraic relation between $r_1, \dots, r_m$, because LLL will always produce at least one short vector. If we were provided in advance with $m-1$ additional relations, carefully chosen to ensure that they would be algebraically independent of the new one, then we would have no need for heuristic assumptions. Unfortunately, we know of no natural occurrences of this case in cryptography, but we will see later in this section that an analogue arises naturally in coding theory, namely in Parvaresh-Vardy codes [21].

The condition $\beta \gg 1/\sqrt{\log N}$ arises from the exponential approximation factor in LLL. It amounts to $N^{\beta^2} \gg 1$. An equivalent formulation is $\log p \gg \sqrt{\log N}$; i.e., the number of digits in the approximate common factor p must be more than the square root of the number of digits in N. When $m = 1$, this is not a restriction at all: when p is small enough that N^{β^2} is bounded, there are only a bounded number of possibilities for r_1 and we can simply try all of them. When $m > 1$, the multivariate algorithm can handle much larger values of r_i for a given p, but the $\log p \gg \sqrt{\log N}$ condition dictates that p cannot be any smaller than when $m = 1$. Given a lattice basis reduction algorithm with approximation factor $2^{(\dim L)^\varepsilon}$, one could replace this condition with $\beta^{1+\varepsilon} \log N \gg 1$. If $\varepsilon = 1/m$, then the constraint can be removed entirely in the m-variable algorithm. See Section 2 for the details.

The $\log p \gg \sqrt{\log N}$ condition is the only thing keeping us from breaking the fully homomorphic encryption scheme from [11]. Specifically, improving the approximation of lattice basis reduction for the particular lattices L we are looking at to $2^{\sqrt{\dim L}}$ would break the suggested parameters in the system. See Section 3 for the details.

We get nearly the same bounds for the "general" approximate common divisor problem, in which we are not given the exact multiple N.

Heuristic Theorem 2 (General approximate common divisors)**.** *Given generic positive integers $a_1, \dots, a_m$ (with $a_i \approx N$ for all i) and bounds $\beta \gg 1/\sqrt{\log N}$ and X, we can find all $r_1, \dots, r_m$ such that*

$$\gcd(a_1 - r_1, \dots, a_m - r_m) \geq N^\beta$$

and $|r_i| \leq X$, provided that

$$X < N^{(C_m + o(1))\beta^{m/(m-1)}},$$

where

$$C_m = \frac{1 - 1/m^2}{m^{1/(m-1)}} \approx 1 - \frac{\log m}{m}.$$

The algorithm runs in polynomial time for fixed m, and the $\gg$ and $o(1)$ are as $N \to \infty$.

Again, for $m = 2$, this result is due to Howgrave-Graham [15].

The proof is very similar to the case when N is known, but the calculations are more tedious because the determinant of the lattice is more difficult to bound. See Section 2.1 for the details.

In [15], Howgrave-Graham gives a more detailed analysis of the behavior for $m = 2$. Instead of our exponent $C_2\beta^2 = \frac{3}{8}\beta^2$, he gets $1 - \beta/2 - \sqrt{1 - \beta - \beta^2/2}$, which is asymptotic to $\frac{3}{8}\beta^2$ for small β but is slightly better when β is large. We are interested primarily in the case when β is small, so we have opted for simplicity, but one could carry out a similar analysis for all m.

1.2.2. *Noisy multi-polynomial interpolation.* Let F be a field. Given m single-variable polynomials $g_1(z), \ldots, g_m(z)$ over F and n distinct points $z_1, \ldots, z_n$ in F, evaluating the polynomials at these points yields mn elements $y_{ij} = g_i(z_j)$ of F.

The noisy multi-polynomial interpolation problem asks for a reconstruction of $g_1, \ldots, g_m$ given the evaluation points $z_1, \ldots, z_n$, degree bounds ℓ_i on g_i, and possibly incorrect values y_{ij}. Stated more precisely: we wish to find all m-tuples of polynomials $(g_1, \ldots, g_m)$ satisfying $\deg g_i \leq \ell_i$, for which there are at least βn values of j such that $g_i(z_j) = y_{ij}$ for all i. In other words, some of the data may have been corrupted, but we are guaranteed that there are at least βn points at which all the values are correct.

This problem is an important stepping stone between single-variable interpolation problems and full multivariate interpolation, in which we reconstruct polynomials of many variables. The multi-polynomial interpolation problem allows us to take advantage of multivariate techniques to prove much stronger bounds, without having to worry about issues such as whether our evaluation points are in general position.

We can restate the noisy multi-polynomial interpolation problem slightly to make the analogy with the integer case clear. Given evaluation points z_j and values y_{ij}, define $N(z) = \prod_j (z - z_j)$, and use ordinary interpolation to find polynomials $f_i(z)$ such that $f_i(z_j) = y_{ij}$. Then we will see shortly that $g_1, \ldots, g_m$ solve the noisy multi-polynomial interpolation problem iff

$$\deg \gcd(f_1(z) - g_1(z), \ldots, f_m(z) - g_m(z), N(z)) \geq \beta n.$$

This is completely analogous to the approximate common divisor problem, with $N(z)$ as the exact multiple and $f_1(z), \ldots, f_m(z)$ as the approximate multiples.

To see why this works, observe that the equation $g_i(z_j) = y_{ij}$ is equivalent to $g_i(z) \equiv y_{ij} \pmod{z - z_j}$. Thus, $g_i(z_j) = f_i(z_j) = y_{ij}$ iff $f_i(z) - g_i(z) \equiv 0 \pmod{z - z_j}$, and $\deg \gcd(f_i(z) - g_i(z), N(z))$ counts how many j satisfy $g_i(z_j) = y_{ij}$. Finally, to count the j such that $g_i(z_j) = y_{ij}$ for all i, we use

$$\deg \gcd(f_1(z) - g_1(z), \ldots, f_m(z) - g_m(z), N(z)).$$

This leads us to our result in the polynomial case.

Heuristic Theorem 3. *Given generic polynomials $N(z), f_1(z), \ldots, f_m(z)$ and degree bounds $\ell_1, \ldots, \ell_m$, we can find all $g_1(z), \ldots, g_m(z)$ such that*

$$\deg \gcd(f_1(z) - g_1(z), \ldots, f_m(z) - g_m(z), N(z)) \geq \beta \deg N(z)$$

and $\deg g_i \leq \ell_i$, provided that

$$\frac{\ell_1 + \cdots + \ell_m}{m} < \beta^{(m+1)/m} \deg N(z).$$

The algorithm runs in polynomial time for fixed m.

As in the integer case, we have given only a heuristic analysis of this algorithm, but it may be easier to make Heuristic Theorem 3 rigorous, because lattice basis

reduction is far more effective and easier to analyze over polynomial rings than it is over the integers.

Parvaresh-Vardy codes [21] are based on noisy multi-polynomial interpolation: a codeword is constructed by evaluating polynomials $f_1, \dots, f_m$ at points $z_1, \dots, z_n$ to obtain mn elements $f_i(z_j)$. In their construction, $f_1, \dots, f_m$ are chosen to satisfy $m-1$ polynomial relations, so that they only need to find one more algebraically independent relation to solve the decoding problem. Furthermore, the $m-1$ relations are constructed so that they must be algebraically independent from the relation constructed by the decoding algorithm. This avoids the need for the heuristic assumption discussed above in the integer case. Furthermore, the Guruswami-Rudra codes [12] achieve improved rates by constructing a system of polynomials so that only n symbols need to be transmitted, rather than mn.

Parvaresh and Vardy gave a list-decoding algorithm using the method of Guruswami and Sudan, which constructs a polynomial by solving a system of equations to determine the coefficients. In our terms, they proved the following theorem:

Theorem 4. *Given a polynomial $N(z)$ and m polynomials $f_1(z), \dots, f_m(z)$, and degree bounds $\ell_1, \dots, \ell_m$, we can find a nontrivial polynomial $Q(x_1, \dots, x_m)$ with the following property: for all $g_1(z), \dots, g_m(z)$ such that*

$$\deg \gcd(f_1(z) - g_1(z), \dots, f_m(z) - g_m(z), N(z)) \geq \beta \deg N(z)$$

and $\deg g_i \leq \ell_i$, we have

$$Q(g_1(z), \dots, g_m(z)) = 0,$$

provided that

$$\frac{\ell_1 + \cdots + \ell_m}{m} < \beta^{(m+1)/m} \deg N(z).$$

The algorithm runs in polynomial time.

In Section 4, we give an alternative proof of this theorem using the analogue of lattice basis reduction over polynomial rings. This algorithm requires neither heuristic assumptions nor conditions on β.

2. Computing approximate common divisors

In this section, we describe our algorithm to solve the approximate common divisor problem over the integers.

To derive Heuristic Theorem 1, we will use the following approach:

(1) Construct polynomials $Q_1, \dots, Q_m$ of m variables such that
$$Q_i(r_1, \dots, r_m) = 0$$
for $r_1, \dots, r_m$ satisfying the conditions of the theorem.
(2) Solve this system of equations to learn candidates for the roots $r_1, \dots, r_m$.
(3) Test each of the polynomially many candidates to see if it is a solution to the original problem.

In the first step, we will construct polynomials Q satisfying

$$Q(r_1, \dots, r_m) \equiv 0 \pmod{p^k}$$

(for a k to be chosen later) whenever $a_i \equiv r_i \pmod{p}$ for all i. We will furthermore arrange that

$$|Q(r_1, \dots, r_m)| < N^{\beta k}.$$

These two facts together imply that $Q(r_1, \dots, r_m) = 0$ whenever $p \geq N^\beta$.

To ensure that $Q(r_1, \ldots, r_m) \equiv 0 \pmod{p^k}$, we will construct Q as an integer linear combination of products

$$(x_1 - a_1)^{i_1} \ldots (x_m - a_m)^{i_m} N^\ell$$

with $i_1 + \cdots + i_m + \ell \geq k$. Alternatively, we can think of Q as being in the integer lattice generated by the coefficient vectors of these polynomials. To ensure that $|Q(r_1, \ldots, r_m)| < N^{\beta k}$, we will construct Q to have small coefficients; i.e., it will be a short vector in the lattice.

More precisely, we will use the lattice L generated by the coefficient vectors of the polynomials

$$(X_1 x_1 - a_1)^{i_1} \ldots (X_m x_m - a_m)^{i_m} N^\ell$$

with $i_1 + \cdots + i_m \leq t$ and $\ell = \max\left(k - \sum_j i_j, 0\right)$. Here t and k are parameters to be chosen later. Note that we have incorporated the bounds $X_1, \ldots, X_m$ on the desired roots $r_1, \ldots, r_m$ into the lattice. We define Q to be the corresponding integer linear combination of $(x_1 - a_1)^{i_1} \ldots (x_m - a_m)^{i_m} N^\ell$, without $X_1, \ldots, X_m$.

Given a polynomial $Q(x_1, \ldots, x_m)$ corresponding to a vector $v \in L$, we can bound $|Q(r_1, \ldots, r_m)|$ by the ℓ_1 norm $|v|_1$. Specifically, if

$$Q(x_1, \ldots, x_m) = \sum_{j_1, \ldots, j_m} q_{j_1 \ldots j_m} x_1^{j_1} \ldots x_m^{j_m},$$

then v has entries $q_{j_1 \ldots j_m} X_1^{j_1} \ldots X_m^{j_m}$, and

$$\begin{aligned} |Q(r_1, \ldots, r_m)| &\leq \sum_{j_1, \ldots, j_m} |q_{j_1 \ldots j_m}| |r_1|^{j_1} \ldots |r_m|^{j_m} \\ &\leq \sum_{j_1, \ldots, j_m} |q_{j_1 \ldots j_m}| X_1^{j_1} \ldots X_m^{j_m} \\ &= |v|_1. \end{aligned}$$

Thus, every vector $v \in L$ satisfying $|v|_1 < N^{\beta k}$ gives a polynomial relation between $r_1, \ldots, r_m$.

It is straightforward to compute the dimension and determinant of the lattice:

$$\dim L = \binom{t+m}{m},$$

and

$$\det L = (X_1 \ldots X_m)^{\binom{t+m}{m} \frac{t}{m+1}} N^{\binom{k+m}{m} \frac{k}{m+1}}.$$

To compute the determinant, we can choose the monomial ordering so that the basis matrix for this lattice is upper triangular; then the determinant is simply the product of the terms on the diagonal.

Now we apply LLL lattice basis reduction to L. Because all the vectors in L are integral, the m shortest vectors $v_1, \ldots, v_m$ in the LLL-reduced basis satisfy

$$|v_1| \leq \cdots \leq |v_m| \leq 2^{(\dim L)/4} (\det L)^{1/(\dim L + 1 - m)}$$

(see Theorem 2 in [14]), and $|v|_1 \leq \sqrt{\dim L}\, |v|$ by Cauchy-Schwarz, so we know that the corresponding polynomials Q satisfy

$$|Q(r_1, \ldots, r_m)| \leq \sqrt{\dim L}\, 2^{(\dim L)/4} (\det L)^{1/(\dim L + 1 - m)}.$$

If

$$\sqrt{\dim L}\, 2^{(\dim L)/4} \det L^{1/(\dim L + 1 - m)} < N^{\beta k}, \tag{2.1}$$

then we can conclude that $Q(r_1, \dots, r_m) = 0$.

If t and k are large, then we can approximate $\binom{t+m}{m}$ with $t^m/m!$ and $\binom{k+m}{m}$ with $k^m/m!$. The $\sqrt{\dim L}$ factor plays no significant role asymptotically, so we simply omit it (the omission is not difficult to justify). After taking a logarithm and simplifying slightly, our desired equation (2.1) becomes

$$\frac{t^m}{4km!} + \frac{1}{1 - \frac{(m-1)!}{t^m}} \left(\frac{m \log X}{m+1} \frac{t}{k} + \frac{\log N}{m+1} \frac{k^m}{t^m} \right) < \beta \log N,$$

where X denotes the geometric mean of $X_1, \dots, X_m$.

The $t^m/(4km!)$ and $(m-1)!/t^m$ terms are nuisance factors, and once we optimize the parameters they will tend to zero asymptotically. We will take $t \approx \beta^{-1/m} k$ and $\log X \approx \beta^{(m+1)/m} \log N$. Then

$$\frac{m \log X}{m+1} \frac{t}{k} + \frac{\log N}{m+1} \frac{k^m}{t^m} \approx \frac{m}{m+1} \beta \log N + \frac{1}{m+1} \beta \log N = \beta \log N.$$

By setting $\log X$ slightly less than this bound (by a $1 + o(1)$ factor), we can achieve the desired inequality, assuming that the $1 - (m-1)!/t^m$ and $t^m/(4km!)$ terms do not interfere. To ensure that they do not, we take $t \gg m$ and $t^m \ll \beta \log N$. In order for this to work, we must have $k \geq 1$; because $k^m \approx \beta t^m$, this means we must have $\beta^2 \log N \gg 1$, and this condition suffices for the algorithm to work. Note that given a lattice basis reduction algorithm with approximation factor $2^{(\dim L)^\varepsilon}$, we could replace t^m with $t^{\varepsilon m}$ and the condition would become $\beta^{1+\varepsilon} \log N \gg 1$.

The final step of the proof is to solve the system of equations defined by the m shortest vectors in the reduced basis to learn $r_1, \dots, r_m$. One way to do this is to repeatedly use resultants to eliminate variables; alternatively, we can use Gröbner bases. See, for example, Chapter 3 of [10].

One obstacle is that the equations may be not algebraically independent, in which case we will not have enough information to complete the solution. In the experiments summarized in Section 6, we sometimes encountered cases when the m shortest vectors were algebraically dependent. However, in every case the vectors represented either (1) irreducible, algebraically independent polynomials, or (2) algebraically dependent polynomials that factored easily into polynomials which all had the desired properties. Thus when the assumption of algebraic dependence failed, it failed because there were fewer than m independent factors among the m shortest relations. In these cases, there were always more than m vectors of ℓ_1 norm less than $N^{\beta k}$, and we were able to complete the solution by using all these vectors. This behavior appears to depend sensitively on the optimization of the parameters t and k.

2.1. **Heuristic Theorem 2.** The algorithm for Heuristic Theorem 2 is identical to the above, except that we do not have an exact N, so we omit all vectors involving N from the construction of the lattice L.

The matrix of coefficients is no longer square, so we have to do more work to bound the determinant of the lattice. Howgrave-Graham [15] observed in the two-variable case that the determinant is preserved even under non-integral row operations, and he used a non-integral transformation to hand-reduce the matrix before bounding the determinant as the product of the ℓ_2 norms of the basis vectors; furthermore, the ℓ_2 norms are bounded by $\sqrt{\dim L}$ times the ℓ_∞ norms.

The non-integral transformation that he uses is based on the relation

$$(x_i - a_i) - \frac{a_i}{a_1}(x_1 - a_1) = x_i - \frac{a_i}{a_1}x_1.$$

By adding a multiple of $f(x)(x_1-a_1)$, one can reduce $f(x)(x_i-a_i)$ to $f(x)(x_i-\frac{a_i}{a_1}x_1)$. The advantage of this is that if $x_1 \approx x_i$ and $a_1 \approx a_i$, then $x_i - \frac{a_i}{a_1}x_1$ may be much smaller than $x_i - a_i$ was. The calculations are somewhat cumbersome, and we will omit the details (see [15] for more information).

When $a_1, \ldots, a_m$ are all roughly N (as in Heuristic Theorem 2), we get the following values for the determinant and dimension in the m-variable case:

$$\det L \leq (N/X)^{\binom{k+m-1}{m}(t-k+1)} X^{m\left(\binom{t+m}{m}\frac{t}{m+1} - \binom{k-1+m}{m}\frac{k-1}{m+1}\right)}$$

and

$$\dim L = \binom{t+m}{m} - \binom{k-1+m}{m}.$$

To optimize the resulting bound, we take $t \approx (m/\beta)^{1/(m-1)}k$.

3. Applications to fully homomorphic encryption

In [11], the authors build a fully homomorphic encryption system whose security relies on several assumptions, among them the hardness of computing an approximate common divisor of many integers. This assumption is used to build a simple "somewhat homomorphic" scheme, which is then transformed into a fully homomorphic system under additional hardness assumptions. In this section, we use our algorithm for computing approximate common divisors to provide a more precise understanding of the security assumption underlying this somewhat homomorphic scheme.

For ease of comparison, we will use their notation (see Section 3 of [11]). Let γ be the bit length of N, η be the bit length of p, and ρ be the bit length of each r_i. Using our algorithm, we can find $r_1, \ldots, r_m$ and the secret key p when

$$\rho \leq \gamma\beta^{(m+1)/m}.$$

Substituting in $\beta = \eta/\gamma$, we obtain

$$\rho^m\gamma \leq \eta^{m+1}.$$

The authors of [11] suggest as a "convenient parameter set to keep in mind" to set $\rho = \lambda$, $\eta = \lambda^2$, and $\gamma = \lambda^5$. Using $m > 3$ we would be able to solve this parameter set, if we did not have the barrier that η^2 must be much greater than γ.

As pointed out in Section 1.2.1, this barrier would no longer apply if we could improve the approximation factor for lattice basis reduction. Given a $2^{\sqrt{\dim L}}$ approximation factor, the condition would be $\beta^{3/2}\gamma \gg 1$, or equivalently $\eta^3 \gg \gamma$. We could then solve the approximate common divisor problem for the recommended parameter setting. In general, if we could achieve an approximation factor of $2^{(\dim L)^\varepsilon}$ for arbitrarily small ε, then we could solve the approximate common divisor problem for parameters given by any polynomials in λ. Furthermore, as we will see in Section 6, the LLL algorithm performs much better in practice on these problems than the theoretical analysis suggests, although we are unable to quantify or predict how much better. However, our experiments suggest that the improved performance at least partially breaks down as γ approaches η^2.

4. Multi-polynomial interpolation

4.1. **Polynomial lattices.** For Heuristic Theorem 3 and Theorem 4, we can use almost exactly the same technique, but with lattices over the polynomial ring $F[z]$ instead of the integers.

By a d-dimensional lattice L over $F[z]$, we mean the $F[z]$-span of d linearly independent vectors in $F[z]^d$. The degree $\deg v$ of a vector v in L is the maximum degree of any of its components, and the determinant $\det L$ is the determinant of a basis matrix (which is well-defined, up to scalar multiplication).

The polynomial analogue of lattice basis reduction produces a basis $b_1, \dots, b_d$ for L such that

$$\deg(b_1) + \cdots + \deg(b_d) = \deg \det L.$$

Such a basis is called a reduced basis (sometimes column or row-reduced, depending on how the vectors are written), and it can be found in polynomial time; see, for example, Section 6.3 in [17]. If we order the basis so that $\deg(b_1) \le \cdots \le \deg(b_d)$, then clearly

$$\deg(b_1) \le \frac{\deg \det L}{d},$$

and more generally

$$\deg(b_i) \le \frac{\deg \det L}{d-(i-1)},$$

because

$$\deg \det L - (d-(i-1))\deg(b_i) = \sum_{j=1}^{d} \deg(b_j) - \sum_{j=i}^{d} \deg(b_i) \ge 0.$$

These inequalities are the polynomial analogues of the vector length bounds in LLL-reduced lattices, but notice that the exponential approximation factor does not occur. See [6] for more information about this analogy.

4.2. **Heuristic Theorem 3 and Theorem 4.** In the polynomial setting, we will choose $Q(x_1, \dots, x_m)$ to be a linear combination (with coefficients from $F[z]$) of the polynomials

$$(x_1 - f_1(z))^{i_1} \dots (x_m - f_m(z))^{i_m} N(z)^{\ell}$$

with $i_1 + \cdots + i_m \le t$ and $\ell = \max\left(k - \sum_j i_j, 0\right)$. We define the lattice L to be spanned by the coefficient vectors of these polynomials, but with x_i replaced with $z^{\ell_i} x_i$ to incorporate the bound on $\deg g_i$, much as we replaced x_i with $X_i x_i$ in Section 2.

As before, we can easily compute the dimension and determinant of L:

$$\dim L = \binom{t+m}{m}$$

and

$$\deg \det L = (\ell_1 + \cdots + \ell_m)\binom{t+m}{m}\frac{t}{m+1} + n\binom{k+m}{m}\frac{k}{m+1},$$

where $n = \deg N(z)$.

Given a polynomial $Q(x_1, \dots, x_m)$ corresponding to a vector $v \in L$, we can bound $\deg Q(g_1(z), \dots, g_m(z))$ by $\deg v$. Specifically, suppose

$$Q(x_1, \dots, x_m) = \sum_{j_1, \dots, j_m} q_{j_1 \dots j_m}(z) x_1^{j_1} \dots x_m^{j_m};$$

then v is the vector whose entries are $q_{j_1\dots j_m}(z)z^{j_1\ell_1+\dots+j_m\ell_m}$, and

$$\begin{aligned}\deg Q(g_1(z),\dots,g_m(z)) &\le \max_{j_1,\dots,j_m} (\deg q_{j_1\dots j_m}(z) + j_1 \deg g_1(z) + \dots + j_m \deg g_m(z)) \\ &\le \max_{j_1,\dots,j_m} (\deg q_{j_1\dots j_m}(z) + j_1\ell_1 + \dots + j_m\ell_m) \\ &= \deg v.\end{aligned}$$

Let $v_1,\dots,v_{\dim L}$ be a reduced basis of L, arranged in increasing order by degree. If

$$\frac{\deg\det L}{\dim L-(m-1)} < \beta kn, \tag{4.1}$$

then each of $v_1,\dots,v_m$ yields a polynomial relation Q_i such that

$$Q_i(g_1(z),\dots,g_m(z)) = 0,$$

because by the construction of the lattice, $Q_i(g_1(z),\dots,g_m(z))$ is divisible by the k-th power of an approximate common divisor of degree βn, while

$$\deg Q_i(g_1(z),\dots,g_m(z)) \le \deg v_i < \beta kn.$$

Thus, we must determine how large $\ell_1+\dots+\ell_m$ can be, subject to the inequality (4.1).

If we set $t \approx k\beta^{-1/m}$ and

$$\frac{\ell_1+\dots+\ell_m}{m} < n\beta^{(m+1)/m},$$

then inequality (4.1) is satisfied when t and k are sufficiently large. Because there is no analogue of the LLL approximation factor in this setting, we do not have to worry about t and k becoming too large (except for the obvious restriction that $\dim L$ must remain polynomially bounded), and there is no lower bound on β. Furthermore, we require no $1+o(1)$ factors, because all degrees are integers and all the quantities we care about are rational numbers with bounded numerators and denominators; thus, any sufficiently close approximation might as well be exact, and we can achieve this when t and k are polynomially large.

5. Higher degree polynomials

It is possible to generalize the results in the previous sections to find solutions of a system of higher degree polynomials modulo divisors of N.

Heuristic Theorem 5. *Given a positive integer N and m generic monic polynomials $h_1(x),\dots,h_m(x)$ over the integers, of degrees $d_1,\dots,d_m$, and given any $\beta \gg 1/\sqrt{\log N}$ and bounds $X_1,\dots,X_m$, we can find all $r_1,\dots,r_m$ such that*

$$\gcd(N, h_1(r_1),\dots,h_m(r_m)) \ge N^\beta$$

and $|r_i| \le X_i$, provided that

$$\sqrt[m]{X_1^{d_1}\dots X_m^{d_m}} < N^{(1+o(1))\beta^{(m+1)/m}}.$$

The algorithm runs in polynomial time for fixed m.

The $m=1$ case is rigorous and encompasses both Howgrave-Graham and Coppersmith's theorems [15, 7]; it first appeared in [20].

When $X_1=\dots=X_m$, the bound becomes $N^{\beta^{(m+1)/m}/\bar{d}}$, where $\bar{d} = (d_1+\dots+d_m)/m$ is the average degree.

Heuristic Theorem 6. *Given a polynomial $N(z)$ and m generic monic polynomials $h_1(x), \dots, h_m(x)$ over $F[z]$, of degrees $d_1, \dots, d_m$ in x, and given degree bounds $\ell_1, \dots, \ell_m$, we can find all $g_1(z), \dots, g_m(z)$ in $F[z]$ such that*

$$\deg \gcd(N(z), h_1(g_1(z)), \dots, h_m(g_m(z))) \geq \beta \deg N(z)$$

and $\deg g_i(z) \leq \ell_i$*, provided that*

$$\frac{\ell_1 d_1 + \cdots + \ell_m d_m}{m} < \beta^{(m+1)/m} \deg N(z).$$

The algorithm runs in polynomial time for fixed m.

The algorithms are exactly analogous to those for the degree 1 cases, except that $x_i - a_i$ (or $x_i - f_i(z)$) is replaced with $h_i(x_i)$.

6. Implementation

We implemented the number-theoretic version of the partial approximate common divisor algorithm using Sage [22]. We used Magma [5] to do the LLL and Gröbner basis calculations.

We solved the systems of equations by computing a Gröbner basis with respect to the lexicographic monomial ordering, to eliminate variables. Computing a Gröbner basis can be extremely slow, both in theory and in practice. We found that it was more efficient to solve the equations modulo a large prime, to limit the bit length of the coefficients in the intermediate and final results. Because $r_1, \dots, r_m$ are bounded in size, we can simply choose a prime larger than $2 \max_i |r_i|$.

We ran our experiments on a Mac Pro with two 2.26 GHz quad-core Intel Xeon processors and 16 GB of RAM. Table 1 shows a selection of sample running times for various parameter settings. For comparison, the table includes the $m = 1$ case, which is Howgrave-Graham's algorithm. The italicized rows give example lattice dimensions for larger inputs to illustrate the limiting behavior of the algorithm; they show the choices of t and k that maximize r given N and p, subject to $\dim L \leq 1000$.

The performance of the algorithm depends on the ratio of t to k, which should be approximately $\beta^{-1/m}$. Incorrectly optimized parameters often perform much worse than correctly optimized parameters. For example, when $m = 3$, $\log_2 N = 1000$, and $\log_2 p = 200$, taking $(t, k) = (4, 2)$ can handle 84-bit perturbations r_i, as one can see in Table 1, but taking $(t, k) = (4, 3)$ cannot even handle 60 bits.

For large m, we experimented with using the non-optimized parameters $(t, k) = (1, 1)$, as reported in Table 1. For the shortest vector only, the bounds would replace the exponent $\beta^{(m+1)/m}$ with $(m + 1)\beta/m - 1/m$, which is its tangent line at $\beta = 1$. This bound is always worse, and it is trivial when $\beta \leq 1/(m + 1)$, but it still approaches the optimal exponent β for large m. Our analysis does not yield a strong enough bound for the m-th largest vector, but in our experiments the vectors found by LLL are much shorter than predicted by the worst-case bounds, so the shortest-vector analysis seems to correctly describe the behavior. Furthermore, the algorithm runs extremely quickly with these parameters, because the lattices have lower dimensions and the simultaneous equations are all linear.

The last column of the table, labeled "LLL factor," describes the approximation ratio obtained by LLL in the experiment. Specifically, LLL factor λ means

$$|v_m| \approx \lambda^{\dim L} (\det L)^{1/(\dim_L - (m-1))},$$

m	$\log_2 N$	$\log_2 p$	$\log_2 r$	t	k	$\dim L$	LLL	Gröbner	LLL factor
1	1000	200	37	55	11	56	148.16s	–	1.02
1	*1000*	*200*	*39*	*166*	*33*	*167*			
1	1000	400	154	40	16	41	87.18s	–	1.01
1	1000	400	156	82	33	83	6312.15s	–	1.03
1	*1000*	*400*	*159*	*248*	*99*	*249*			
2	1000	200	72	9	4	55	40.60s	0.17s	0.85
2	*1000*	*200*	*85*	*40*	*18*	*861*			
2	1000	400	232	10	6	66	196.38s	0.56s	0.70
2	1000	400	238	17	11	171	122624.24s	3.48s	0.94
2	*1000*	*400*	*247*	*41*	*26*	*903*			
3	1000	200	87	5	3	56	28.06s	0.42s	0.79
3	*1000*	*200*	*106*	*16*	*9*	*969*			
3	1000	400	255	4	3	35	4.64s	4.15s*	0.25
3	1000	400	268	7	5	120	2748.80s	6.93s	0.84
3	*1000*	*400*	*281*	*16*	*12*	*969*			
4	1000	200	94	3	2	35	2.19s	0.30s	0.49
4	*1000*	*200*	*116*	*9*	*6*	*715*			
4	1000	400	279	4	3	70	57.12s	4.61s	0.61
4	*1000*	*400*	*296*	*9*	*7*	*715*			
5	1000	200	108	3	2	56	11.62s	3.33s	0.71
5	1000	200	110	4	3	126	1250.70s	144.35s	0.93
5	1000	400	278	3	2	56	2.81s	2.81s	0.34
6	1000	200	115	3	2	84	52.90s	82.37s	0.85
6	1000	400	297	3	2	84	5.23s	2.59s*	0.375
7	1000	200	114	3	2	120	189.93s	2248.35s	0.92
7	1000	400	311	3	2	120	14.37s	5.64s*	0.43
12	1000	400	347	1	1	13	0.01s	1.00s	2.7×10^{-51}
18	1000	400	364	1	1	19	0.03s	0.01s	3.1×10^{-54}
24	1000	400	372	1	1	25	0.05s	3.81s	1.1×10^{-55}
48	1000	400	383	1	1	49	0.37s	17.16s	7.97×10^{-58}
96	1000	400	387	1	1	97	2.69s	62.37s	1.09×10^{-58}

TABLE 1. Experimental results from our implementation of the integer partial approximate common divisor algorithm, with sample parameters for more extreme calculations in italics.

where v_m is the m-th smallest vector in the LLL-reduced basis for L. Interestingly, it is below 1 in almost all of the examples in the table, which indicates that the bounds used earlier in the paper are unduly pessimistic, at least for these parameter sizes. This means larger values of r can be solved for than one might naively expect from our theoretical analysis. Unfortunately, this improvement appears to break down when p becomes small. For example, with $m = 5$, $\log_2 N = 2^{10}$, $\log_2 p = 2^5$, and $\log_2 r = 4$, taking $(t, k) = (4, 1)$ yields an LLL factor of 1.04, and this value

seems to be typical when $\log_2 p \approx \sqrt{\log_2 N}$. It is therefore unlikely that the LLL algorithm will work past the $\log p \gg \sqrt{\log N}$ barrier.

We marked cases where we encountered algebraically dependent relations with an asterisk in Table 1. In each case, we were still able to solve the system of equations by including more relations from the lattice (up to ℓ_1 norm less than $N^{\beta k}$) and solving this larger system.

Acknowledgements

We thank Chris Umans and Alex Vardy for suggesting looking at Parvaresh-Vardy codes. N.H. would like to thank MIT, CSAIL, and Microsoft Research New England for their hospitality during the course of this research. This material is based upon work supported by an AT&T Labs Graduate Fellowship and by the National Science Foundation under Award No. DMS-1103803.

References

[1] D. Augot and M. Finiasz. A public key encryption scheme based on the polynomial reconstruction problem. In *Advances in Cryptology – EUROCRYPT 2003*, pages 229–240. Lecture Notes in Computer Science 2656. Springer-Verlag, Berlin, Heidelberg, 2003.

[2] D. Bleichenbacher, A. Kiayias, and M. Yung. Decoding of interleaved Reed-Solomon codes over noisy data. In *Automata, Languages and Programming*, pages 97–108. Lecture Notes in Computer Science 2719. Springer-Verlag, Berlin, Heidelberg, 2003.

[3] D. Boneh and G. Durfee. Cryptanalysis of RSA with private key d less than $N^{0.292}$. *IEEE Transactions on Information Theory* 46: 1339–1349, 2000.

[4] D. Boneh, G. Durfee, and N. Howgrave-Graham. Factoring $n = p^r q$ for large r. In *Advances in Cryptology – CRYPTO '99*, pages 326–337. Lecture Notes in Computer Science 1666. Springer-Verlag, Berlin, Heidelberg, 1999.

[5] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system I: The user language. *Journal of Symbolic Computation* 24: 235–265, 1997.

[6] H. Cohn and N. Heninger. Ideal forms of Coppersmith's theorem and Guruswami-Sudan list-decoding. In *Proceedings of Innovations in Computer Science*, pages 298–308, 2011. Full version available as `arXiv:1008.1284`.

[7] D. Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology* 10: 233–260, 1997.

[8] D. Coppersmith and M. Sudan. Reconstructing curves in three (and higher) dimensional space from noisy data. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing* (San Diego, CA, USA, June 9–11, 2003), pages 136–142. ACM, New York, NY, 2001.

[9] J. S. Coron. Cryptanalysis of a public-key encryption scheme based on the polynomial reconstruction problem. In *Public Key Cryptography – PKC 2004*, pages 14–27. Lecture Notes in Computer Science 2947. Springer-Verlag, Berlin, Heidelberg, 2004.

[10] D. A. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, New York, 1992.

[11] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology – EUROCRYPT 2010*, pages 24–43. Lecture Notes in Computer Science 6110. Springer-Verlag, Berlin, Heidelberg, 2010. Longer version available as Report 2009/616 in the Cryptology ePrint Archive (`http://eprint.iacr.org/2009/616`).

[12] V. Guruswami and A. Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory* 54: 135–150, 2008.

[13] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Transactions on Information Theory* 45: 1757–1767, 1999.

[14] M. Herrmann and A. May. Solving linear equations modulo divisors: On factoring given any bits. In *Advances in Cryptology – ASIACRYPT 2010*, pages 406–424. Lecture Notes in Computer Science 5350. Springer-Verlag, Berlin, Heidelberg, 2008.

[15] N. Howgrave-Graham. Approximate integer common divisors. In *Cryptography and Lattices*, pages 51–66. Lecture Notes in Computer Science 2146. Springer-Verlag, Berlin, Heidelberg, 2001.
[16] C. S. Jutla. On finding small solutions of modular multivariate polynomial equations. In *Advances in Cryptology – EUROCRYPT '98*, pages 158–170. Lecture Notes in Computer Science 1403. Springer-Verlag, Berlin, Heidelberg, 1998.
[17] T. Kailath. *Linear Systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, 1980.
[18] A. Kiayias and M. Yung. Polynomial reconstruction based cryptography (a short survey). In *Selected Areas in Cryptography*, pages 129–133. Lecture Notes in Computer Science 2259. Springer-Verlag, Berlin, Heidelberg, 2001.
[19] A. Kiayias and M. Yung. Secure games with polynomial expressions. In *Automata, Languages and Programming*, pages 939–950. Lecture Notes in Computer Science 2076. Springer-Verlag, Berlin, Heidelberg, 2001.
[20] A. May. New RSA vulnerabilities using lattice reduction methods. Ph.D. Thesis, University of Paderborn, 2003.
[21] F. Parvaresh and A. Vardy. Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science* (Pittsburgh, PA, USA, October 23–25, 2005), pages 285–294. IEEE Computer Society, Los Alamitos, CA, 2005.
[22] W. A. Stein et al. *Sage Mathematics Software (Version 4.6.2)*. The Sage Development Team, 2011. `http://www.sagemath.org`.

Microsoft Research New England, One Memorial Drive, Cambridge, MA 02142
E-mail address: `cohn@microsoft.com`

Department of Computer Science, Princeton University, Princeton, NJ 08540
Current address: Department of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Drive, Mail Code 0404, La Jolla, CA 92093-0404
E-mail address: `nadiah@cs.ucsd.edu`