Solving Multivariate Coppersmith Problems with Known Moduli

Keegan Ryan

University of California, San Diego, La Jolla CA 92093, USA
kryan@ucsd.edu

Abstract. We examine the problem of finding small solutions to systems of modular multivariate polynomials. While the case of univariate polynomials has been well understood since Coppersmith's original 1996 work, multivariate systems typically rely on carefully crafted shift polynomials and significant manual analysis of the resulting Coppersmith lattice. In this work, we develop several algorithms that make such hand-crafted strategies obsolete. We first use the theory of Gröbner bases to develop an algorithm that provably computes an optimal set of shift polynomials, and we use lattice theory to construct a lattice which provably contains all desired short vectors. While this strategy is usable in practice, the resulting lattice often has large rank. Next, we propose a heuristic strategy based on graph optimization algorithms that quickly identifies low-rank alternatives. Third, we develop a strategy which symbolically precomputes shift polynomials, and we use the theory of polytopes to polynomially bound the running time. Like Meers and Nowakowski's automated method, our precomputation strategy enables heuristically and automatically determining asymptotic bounds. We evaluate our new strategies on over a dozen previously studied Coppersmith problems. In all cases, our unified approach achieves the same recovery bounds in practice as prior work, even improving the practical bounds for three of the problems. In five problems, we find smaller and more efficient lattice constructions, and in three problems, we improve the existing asymptotic bounds. While our strategies are still heuristic, they are simple to describe, implement, and execute, and we hope that they drastically simplify the application of Coppersmith's method to systems of multivariate polynomials.

Keywords: Coppersmith's method, Shift polynomials, Gröbner bases

1 Introduction

Coppersmith's method of finding small roots of polynomial equations is one of the most widely applied techniques in algebraic cryptanalysis. Although originally developed to find a root of a single univariate modular polynomial, it has been repeatedly adapted to heuristically find roots of multivariate polynomials, integer polynomials, and polynomials modulo divisors. Coppersmith's method is a powerful tool, but the main obstacle is that it involves significant manual

analysis and algorithm design for every new system of polynomials one wishes to solve. In this paper, we analyze systems of multivariate polynomials with known moduli (or known multiples of moduli), and we develop a collection of proven and heuristic results that all but eliminate the previously required manual labor.

The challenging part of Coppersmith's method is selecting shift polynomials. In essence, the coefficients of a polynomial f modulo p define a linear combination of its monomials. When a root is small, the monomial valuations are small, and lattice reduction can be used to efficiently find a small solution to the linear relations. However, lattice reduction performs worse the more monomials are involved. In [13], Coppersmith realized that the coefficients of the shifted polynomial xf induce an independent linear constraint, while only introducing a single new monomial. By considering shift polynomials of the form $x^i f^j$, Coppersmith showed that the benefit of adding shift polynomials outweighs the cost of introducing monomials so long as the desired root is smaller than $p^{1/\deg f}$. This analysis of overlapping monomials in shift polynomials is easy in the univariate case, but challenging in the multivariate case.

If one has a system $\mathcal{F}$ of multivariate polynomials with a shared root, each polynomial still induces a linear relation on the monomials, but the amount of monomial overlap in, for example, $x_1^{i_1} x_2^{i_2} f_1^{j_1} f_2^{j_2}$ depends significantly on the precise monomials in $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$. This is why so much manual analysis is necessary for every different system $\mathcal{F}$. One must carefully design a strategy for constructing shift polynomials with sufficient monomial overlap for lattice reduction to recover a small root, and then one must analyze the asymptotic behavior of their strategy to find the largest bound on the small root for which their method succeeds.

1.1 Our Contributions

We present a collection of tools and techniques that automate the multivariate Coppersmith method for systems of modular polynomials.[1] Our contributions eliminate much of the arduous manual work that previously went into constructing shift polynomials and analyzing their performance, and our algorithms are fully practical. In order to achieve this, we rely on results from graph theory, computer algebra, and discrete geometry. Like previous multivariate Coppersmith approaches, our full algorithms are heuristic, but many of the intermediate results are proven rigorously.

In Section 2, we give general background on multivariate Coppersmith problems, and in Section 3, we revisit the connection between shift polynomials and polynomial ideals. This connection has been observed previously, but we develop it further and show that all shift polynomial selection strategies (to our knowledge) involve constructing polynomials that belong to particular ideals.

In Section 4, we give a simple, novel, and provable strategy to select shift polynomials from the ideal. This strategy uses the theory of Gröbner bases over

[1] Our implementation is available at https://github.com/keeganryan/cuso

Euclidean domains, and we prove that our resulting monomials and shift polynomials enable us to build a lattice that is guaranteed to have shorter vectors than could be obtained via any other choice of shift polynomials.

In Section 5, we explore a heuristic strategy to automatically identify low-rank and dense sublattices based on the coefficients of the optimal shift polynomials. This significantly reduces the cost of lattice reduction. Our heuristic strategy is based on an optimization technique for weighted directed graphs, and in practice, it is fast and highly effective.

In Section 6, we describe a precomputation strategy that eliminates the cost of Gröbner basis computations from shift polynomial selection. While this strategy lacks some of the provable guarantees of our prior strategies and requires manually identifying good precomputation parameters, these simplifications allow us to apply the theory of Ehrhart polynomials to automatically determine the asymptotic bounds for a given multivariate Coppersmith problem.

Finally, in Section 7 we describe experiments on 14 different Coppersmith problems to demonstrate the effectiveness of our new approach. We include a wide variety of problems which exhibit our algorithm's ability to match even the most advanced shift polynomial strategies of prior work. In all cases, our unified approach achieves similar practical bounds, often outperforming prior work, all while requiring minimal problem-specific configuration.

2 Background

In this work, we frequently consider polynomials in $\mathbb{Z}[x_1, \ldots, x_\ell]$. Polynomials consist of *terms* that are monomials $m = \prod_{i=1}^{\ell} x_i^{e_i}$ multiplied by coefficients c_m. For a particular *monomial ordering*, each nonzero polynomial f has a leading term $\mathrm{LT}(f)$ with leading monomial $\mathrm{LM}(f)$ and leading coefficient $\mathrm{LC}(f)$. For brevity, we often use vector notation: $f(\boldsymbol{x})$ is shorthand for $f(x_1, \ldots, x_\ell)$. We are also sometimes casual with function notation: for example, $\mathrm{LM}(\mathcal{S})$ means the set of leading monomials of polynomials in the set $\mathcal{S}$.

2.1 Multivariate Coppersmith Problems

Definition 1 (Multivariate Coppersmith problem). *A multivariate Coppersmith problem in ℓ variables involves finding the set of bounded, common roots of a system of modular or integer polynomials. Each input polynomial f_i in a set $\mathcal{F} \subset \mathbb{Z}[\boldsymbol{x}]$ is associated with a constraint on the evaluation of f_i at a potential solution $\boldsymbol{r} \in \mathbb{Z}^\ell$. This constraint is either modular or integer, and each modulus p_i may either be known or bounded below by a known value:*

$$f_i(\boldsymbol{r}) \equiv 0 \pmod{p_i} \quad \text{or} \quad f_i(\boldsymbol{r}) = 0$$

For bounds $\boldsymbol{X} \in \mathbb{Z}^\ell$, find all small points $\boldsymbol{r} \in \mathbb{Z}^\ell$ with $|r_i| < X_i$ for $i = 1, \ldots, \ell$ such that $\boldsymbol{r}$ satisfies the constraint associated with the polynomial $f_i \in \mathcal{F}$.

We use the term *relation* to refer to the combination of a polynomial in $\mathbb{Z}[\boldsymbol{x}]$ and its associated constraint. This definition is more generic than in prior work, since it allows for combinations of input relations with different moduli or no moduli at all. Although this definition includes integer Coppersmith problems, our focus in this work is multivariate problems with at least one modular constraint and where a multiple N_i of each modulus p_i is known. This includes many interesting applications of Coppersmith's method. Existing approaches almost all follow the same general outline:

1. Combine polynomials in $\mathcal{F}$ to generate a set of *shift polynomials*.
2. Construct a lattice basis using the shift polynomials.
3. Run a lattice reduction algorithm to obtain a reduced basis.
4. Use short vectors in the basis to recover the set of bounded roots.

Steps 2, 3, and 4 are well understood at this point and vary little between applications. However, step 1 is where all of the challenge lies.

2.2 Shift Polynomial Selection

Coppersmith's original work considered systems of a single, univariate f with known modulus p [13]. All shift polynomials are of the form $x^i f^j$, and Coppersmith showed that lattice reduction succeeds for this selection of shift polynomials when $\log_p X \lesssim 1/\deg f$, which is provably optimal [11]. May gave a generalization of Coppersmith's result where p is unknown, but a multiple N of p is known [28]. Coppersmith observed that the same shift polynomials and lattice methods may work in principle for multivariate polynomials, but did not explore this in depth.

Jochemsz and May considered systems of a single, multivariate f with known modulus p (and also the integer variant) [22]. They describe a shift polynomial strategy based on the monomial sets constructed by considering monomials in $\boldsymbol{x}^{\boldsymbol{t}} f^k$. In Appendix A, they show how their generalized and heuristic strategy reproduces the same bounds for problems studied by Boneh and Durfee [7] and Blömer and May [5].

More recently, Meers and Nowakowski studied systems of multiple multivariate polynomials with known modulus p [33]. They describe a heuristic strategy for choosing a set of monomials and give an algorithm for selecting shift polynomials from the set of monomials. They claim that their method is globally optimal, but we show that it is not.

Although it has long been a goal to develop a truly generalized strategy for selecting shift polynomials, it is far more common in the literature to find problem-specific strategies. Developing and analyzing increasingly intricate shift polynomial strategies is time-consuming, so the maximum recoverable bounds for a given problem take years to grow. Take for example the Modular Inversion Hidden Number Problem (MIHNP) from Boneh et al. in 2001 [8] which studies ℓ relations of the form $f_i(\alpha, \boldsymbol{x}) = \alpha x_i + c_{i,1}\alpha + c_{i,2}x_i + c_{i,3} \pmod p$. They give concrete and asymptotic shift polynomial strategies that succeed

for $\log_p X_i < 1/3$ and $2/3$. Better strategies were proposed in 2014 [44] and 2018 [46], culminating in the breakthrough result by Xu et al. at Crypto 2019 [47] that $\log_p X_i < \ell/(\ell+1) - o(1/\ell)$ is heuristically solvable. The results were refined further in 2023 [48], but the approach is still limited in practice, since $\log_p X_i < 0.669$ requires $\ell = 32$ and a lattice of dimension 46441.

Similar incremental improvements in shift polynomial strategies appear for the Elliptic Curve Hidden Number Problem [45,49] and RSA-CRT with small private exponents [26,4,23,20,43], not to mention the many RSA partial key exposure variants [16,1,40,42,30,31]. These improvements demonstrate that the existing generalized multivariate shift-polynomial strategies are insufficient for maximizing the recoverable bounds using Coppersmith's method.

2.3 Lattice Basis Construction

Given a set of shift polynomials $\mathcal{S}$ where all nonzero terms involve monomials in a set $\mathcal{M}$, there are a couple of ways to build the lattice. Coppersmith's original work [13] builds a lattice in which vectors represent solutions to a linearized version of $\mathcal{S}$. We call this the *primal lattice*. For ease of explanation, we will focus on Howgrave-Graham's construction of the *dual lattice* [21], in which vectors represent polynomials. If all $f \in \mathcal{S}$ share a bounded root modulo p, then a basis of the dual lattice $\Lambda_{\mathcal{S}}$ is given by

$$\left\{ (c_m m(\boldsymbol{X}))_{m \in \mathcal{M}} \mid f(\boldsymbol{x}) = \sum_{m \in \mathcal{M}} c_m m(\boldsymbol{x}) \in \mathcal{S} \right\}.$$

For example, the Howgrave-Graham lattice of $\mathcal{S} = \{N, Nx_1, Nx_2, x_1x_2 + ax_1 + b, x_1x_2^2 + ax_1x_2 + bx_2\}$ is spanned by the rows of the basis matrix

$$\begin{pmatrix} N & 0 & 0 & 0 & 0 \\ 0 & NX_1 & 0 & 0 & 0 \\ 0 & 0 & NX_2 & 0 & 0 \\ b & aX_1 & 0 & X_1X_2 & 0 \\ 0 & 0 & bX_2 & aX_1X_2 & X_1X_2^2 \end{pmatrix}.$$

The vectors in the lattice correspond to (scaled coefficient vectors of) polynomials. When $\mathcal{S}$ sorted by a monomial order yields a triangular basis (also known as suitability), the dual lattice is full rank, has dimension $|\mathcal{M}|$, and has determinant

$$\det \Lambda_{\mathcal{S}} = \prod_{f \in \mathcal{S}} \mathrm{LT}(f)(\boldsymbol{X}).$$

The primal lattice is useful for integer constraints or when the modulus is known. The dual lattice is useful when the shift polynomials share a common modular constraint. When the modulus is known, Howgrave-Graham showed that the primal and dual Coppersmith constructions are related by lattice duality [21]. This work focuses on modular Coppersmith problems with a known multiple of the modulus, so we will use the dual construction.

2.4 Lattice Reduction

Given a basis for lattice Λ of rank d, a lattice reduction algorithm outputs a *reduced* basis $(\boldsymbol{b}'_1, \ldots, \boldsymbol{b}'_d)$ of short, nearly orthogonal vectors. The LLL algorithm [24] runs in polynomial time, and the basis vectors in the output satisfy

$$\|\boldsymbol{b}'_i\| \leq 2^{\frac{d(d-1)}{4(d+1-i)}} \det \Lambda^{1/(d+1-i)} \quad \text{and} \quad \|\boldsymbol{b}'_i\| \leq 2^{d-1}\lambda_i(\Lambda)$$

where $\lambda_i(\Lambda)$ is the i^{th} *successive minimum* of the lattice, or the minimum radius of a ball at the origin containing i linearly independent vectors. These bounds are found in [27] and [35], and analogous bounds can be derived for more modern reduction algorithms capable of reducing lattices of large rank [38].

2.5 Root Recovery

In the dual lattice, a short vector corresponds to a polynomial g with small coefficients. g is an integer linear combination of polynomials in $\mathcal{S}$, so it satisfies the same constraint (for example $g(\boldsymbol{r}) \equiv 0 \pmod{p}$). The following result of Håstad [18] and Howgrave-Graham [21] is used to show that $g(\boldsymbol{r}) = 0$ over the integers, not just modulo p. We refer to this as the HHG bound.[2]

Lemma 1 (Håstad/Howgrave-Graham). *For a dual Coppersmith lattice of dimension n and bound $\boldsymbol{X}$, let $\boldsymbol{v}$ be a vector and $g \in \mathbb{Z}[\boldsymbol{x}]$ be the corresponding polynomial. If $g(\boldsymbol{r}) \equiv 0 \pmod{p}$ for $|\boldsymbol{r}| < \boldsymbol{X}$ and $\|\boldsymbol{v}\| < p/\sqrt{n}$, then $g(\boldsymbol{r}) = 0$.*

To recover the small root, we hope to find at least ℓ polynomials satisfying this bound. These polynomials share a common root over the integers, and the following heuristic is used to conclude that the root can be found, using Gröbner bases for example.

Heuristic 1 *The algebraic variety corresponding to the ideal in $\mathbb{Q}[\boldsymbol{x}]$ of polynomials recovered by lattice reduction is zero-dimensional.*

When $\mathcal{S}$ is suitable and all polynomials in $\mathcal{S}$ share a root modulo p, the following condition describes when Coppersmith's method heuristically succeeds.

$$2^{\frac{|\mathcal{S}|(|\mathcal{S}|-1)}{4}} \prod_{f \in S} \mathrm{LT}(f)(\boldsymbol{X}) < \left(\frac{p}{\sqrt{|\mathcal{S}|}}\right)^{|\mathcal{S}|+1-\ell} \tag{1}$$

This condition on $\mathcal{S}$ ensures that the determinant bound for lattice reduction implies recovery of ℓ linearly independent vectors satisfying the HHG bound. It is common to see this in an asymptotic form, where we consider arbitrarily large p and $|\mathcal{S}|$ such that the contribution of several terms becomes negligible.

$$\prod_{f \in S} \mathrm{LT}(f)(\boldsymbol{X}) < p^{|\mathcal{S}|} \tag{2}$$

[2] This is often referred to as the Howgrave-Graham bound, but as May notes [29], it appears in Håstad's earlier work as well.

2.6 On the Possibility of an Efficient and Provable Algorithm

In [14], Coppersmith showed that an efficient and provable algorithm for solving multivariate Coppersmith problems is not possible. More precisely, he shows that finding small solutions to the modular equation $ax_1^2 + bx_2 - c \equiv 0 \pmod{p}$ implies an efficient solution to an NP-complete problem from number theory.

Despite this inherent limitation, there has been great progress in solving multivariate problems by considering non-polynomial-time subroutines and making heuristic assumptions. For example, Gröbner basis computation has doubly exponential worst-case running time, but modern computer algebra libraries are efficient, and computing Gröbner bases is usually not a bottleneck in practice. In line with previous work, our focus is also on developing a fast heuristic approach which is well supported by practical experiments.

3 Ideals

In this section, we explore how shift polynomials can be represented as members of an ideal in $\mathbb{Z}[\boldsymbol{x}]$. Prior work has also shown the connection between shift polynomials and ideals [2], but we hope to elaborate on this in greater depth.

Ideals in a ring represent combinations of the ideal's generators $\mathcal{F} \subset \mathbb{Z}[\boldsymbol{x}]$:

$$J = \langle \mathcal{F} \rangle = \{ \sum_i a_i f_i \mid a_i \in \mathbb{Z}[\boldsymbol{x}], f_i \in \mathcal{F} \}.$$

This resembles the common shift polynomial strategy of multiplying input polynomials by monomials: if f has a root modulo p, then $x^j f$ has the same root modulo p. Indeed, if the generators of an ideal share a root modulo p, then so do all the elements in the ideal. Addition and multiplication are defined for ideals:

$$J + J' = \{ f + f' \mid f \in J, f' \in J' \} \quad J \cdot J' = \langle \{ f f' \mid f \in J, f' \in J' \} \rangle.$$

Multiplication resembles the common shift polynomial strategy of *multiplicities*. If N and f share a root modulo p, then N^2, Nf, and f^2 share a root modulo p^2, have multiplicity 2, and belong to the ideal $\langle N, f \rangle^2$. Ideals of this form are considered in [2]. If polynomials in J share a root modulo p and polynomials in J' share the same root modulo p', then polynomials in $J + J'$ share the root modulo $\gcd(p, p')$, and polynomials in $J \cdot J'$ share the root modulo pp'.

In general, any sort of shift polynomial strategy that involves taking polynomial combinations input polynomials can be represented as finding members of an ideal. This encompasses all shift polynomial strategies we are aware of, including the linear algebra–based strategy in [46,47,48], the exponent tricks in [25] or the technique of unravelled linearization [19].

3.1 Unravelled Linearization

In 2009, Herrmann and May proposed a novel technique for shift polynomial selection called unravelled linearization [19]. In essence, they observed that in

the specific input relation $f(\boldsymbol{x}) = x_1^2 - x_2 + ax_1 + b \equiv 0 \pmod{p}$, it helps to group together ("linearize") the terms $(x_1^2 - x_2) \mapsto u$ into a new bounded variable, so $g(\boldsymbol{x}, u) = u + ax_1 + b \equiv 0 \pmod{p}$. Next, they calculate polynomials of the form $g_{i,j} = x_1^j g^i$, and finally they back-substitute ("unravel") $x_1^2 \mapsto u + x_2$ into $g_{i,j}$ to eliminate all monomials that are a multiple of x_1^2. This decreases the resulting lattice determinant and increases the power of the attack.

There is a simple way of representing unravelled linearization with ideals. We introduce a new variable u and new polynomial $f_{\text{ul}}(x_1, x_2, u) = x_1^2 - u - x_2$. Now f, f_{ul}, and p all share a small root modulo p bounded by $(X_1, X_2, X_1^2 + X_2)$, so this is a multivariate Coppersmith problem. The process of unravelling involves subtracting polynomial multiples of f_{ul}, so the resulting shift polynomial $g_{i,j}$ is in the ideal $\langle f, p\rangle^i + \langle f_{\text{ul}}\rangle$. If we have a monomial ordering where $u < x_1^2$, then reduction by $\langle f_{\text{ul}}\rangle$ corresponds to eliminating all monomials that are a multiple of x_1^2. We essentially perform unravelled linearization by augmenting the multivariate Coppersmith problem with an additional polynomial $f_{\text{ul}} \in \mathcal{F}'$ with an integer constraint.

3.2 Determining the Shift Polynomial Ideal

For input polynomials $\mathcal{F}$, the modular or integer constraints, and root $\boldsymbol{r}$, define

$$\hat{J}_{p_i} = \langle \{ f \in \mathcal{F} \mid f(\boldsymbol{r}) \equiv 0 \pmod{p_i} \} \rangle \quad \text{and} \quad \hat{J}_\infty = \langle \{ f \in \mathcal{F} \mid f(\boldsymbol{r}) = 0 \} \rangle .$$

Note that if p_i or a multiple N_i of p_i is known, then $p_i \equiv 0 \pmod{p_i}$ or $N_i \equiv 0 \pmod{p_i}$ are implicit input relations described by constant polynomials in $\mathcal{F}$. If all modular relations share the same modulus p, then we may select multiplicity k and define the ideal

$$J_k = \hat{J}_p^k + \hat{J}_\infty$$

which has the property that for $f \in J_k$, $f(\boldsymbol{r}) \equiv 0 \pmod{p^k}$.

However, we also consider situations where the relations involve multiple moduli [25,32]. Let P be the set of distinct moduli and let Q be a set of pairwise coprime divisors of the p_i. For the sake of notation, we treat ∞ as a modulus and say everything divides ∞. Then for multiplicity $\boldsymbol{k} \in \mathbb{Z}_{\geq 0}^{|Q|}$, we define

$$J_{\boldsymbol{k}} = \sum_{\boldsymbol{e} \in E_{\boldsymbol{k}}} \prod_i \hat{J}_{p_i}^{e_i} \quad \text{where} \quad E_{\boldsymbol{k}} = \left\{ \boldsymbol{e} \in \mathbb{Z}_{\geq 0}^{|P|} \middle| \prod_j q_j^{k_j} \text{ divides } \prod_i p_i^{e_i} \right\} .$$

Observe that if $\boldsymbol{r}$ satisfies the constraints for the input polynomials, then it also satisfies the polynomial $f \in \prod_i \hat{J}_{p_i}^{e_i}$ with the constraint $f(\boldsymbol{r}) \equiv 0 \pmod{\prod_i p_i^{e_i}}$ for all $\boldsymbol{e} \in E_{\boldsymbol{k}}$ and the polynomial $f \in J_{\boldsymbol{k}}$ with the constraint $f(\boldsymbol{r}) \equiv 0 \pmod{\prod_j q_j^{k_j}}$ for all $\boldsymbol{k}$. One can efficiently compute $J_{\boldsymbol{k}}$ from smaller multiplicities $J_{\boldsymbol{k}'}$ using dynamic programming. Also note that this definition agrees with the previous one for $P = \{p, \infty\}$ and $Q = \{p\}$.

4 Optimal Shift Polynomial Selection

We can construct the ideal which contains all shift polynomials of a given multiplicity, but it remains an important question how to select the shift polynomials for inclusion in the dual lattice. In this section, we provide a provably optimal strategy that requires selecting a set of monomials in advance. While the main focus of this paper is modular Coppersmith problems, we note that this strategy applies equally well to integer Coppersmith problems.

The concept of constructing shift polynomials from a preselected set of monomials is a common one. This framework was used by Jochemsz and May in 2006 [22], and more recently by Meers and Nowakowski in 2023 [33]. The automated method of Meers and Nowakowski involves taking input polynomials and finding some product of their leading monomials that divides each monomial in the preselected set $\mathcal{M}$. They claim that their method finds the optimal set of shift polynomials. We will show that, even for their example application, this is not the case.

4.1 Gröbner Bases over Euclidean Domains

Our algorithm is based on the theory of Gröbner bases over principal ideal domains and Euclidean domains. For our purposes, this is used to find Gröbner bases for ideals in $\mathbb{Z}[\boldsymbol{x}]$. While there are some crucial differences compared to the more familiar case of ideals in $\mathbb{Q}[\boldsymbol{x}]$, many of the properties of Gröbner bases over fields have analogues in our setting.

Gröbner bases have been used many times before in Coppersmith-like problems, but not in this way. As discussed in Section 2.5, Gröbner bases for ideals in $\mathbb{Q}[\boldsymbol{x}]$ are frequently used to find the small root following lattice reduction. Some more recent works replace this ring with $\mathbb{F}_p[\boldsymbol{x}]$ and use the Chinese remainder theorem to reconstruct the zero-dimensional variety [33,48,49]. Herrmann and May also used Gröbner bases in 2010 to find a nontrivial relationship between unravelled linearization variables [20].

Gröbner bases are defined relative to a monomial ordering. There are many possible orderings, but we typically use the weight order defined by the bounds on the unknown variables. If small roots are bounded by $|\boldsymbol{r}| < \boldsymbol{X}$, then we order monomials by $<_{\boldsymbol{X}}$, where $m_1 <_{\boldsymbol{X}} m_2$ if $m_1(\boldsymbol{X}) < m_2(\boldsymbol{X})$. In case of ties, we fall back to lexicographic order.[3]

The textbook of Becker and Weispfenning [3] describes an algorithm to find what they call a *D-Gröbner basis* G of an ideal $J \in R[\boldsymbol{x}]$ where R is a principal ideal domain. G is a finite subset of J that generates the ideal. We rely on this additional property of D-Gröbner bases, adapted from [3, Exercise 10.5].

Lemma 2 (Becker and Weispfenning). *Let G be a D-Gröbner basis of an ideal $J \in R[\boldsymbol{x}]$. For nonzero $f \in J$, there exists $g \in G$ where* $\operatorname{LT}(g) | \operatorname{LT}(f)$.

[3] We take advantage of this tie-breaking when using unravelled linearization. For the example in Section 3.1, a weight order of $(\log X_1, \log X_2, 2\log X_1)$ will combine with the tie-break behavior to achieve $u < x_1^2$ as desired.

To illustrate the difference between Gröbner bases over fields and D-Gröbner bases, consider the polynomials $f_1(x) = 10$ and $f_2(x) = 3x^2 + 7$. The ideal $\langle f_1, f_2 \rangle \in \mathbb{Q}[x]$ is trivial, because $\frac{1}{10} f_1 = 1$ is in the ideal, so the Gröbner basis is simply $\{1\}$. However, since 10 is not invertible in $\mathbb{Z}$, the D-Gröbner basis of ideal $\langle f_1, f_2 \rangle \in \mathbb{Z}[x]$ is $\{x^2 + 9, 10\}$. We note that $x^2 + 9 \equiv 3^{-1}(3x^2 + 7) \pmod{10}$, so the process of computing the D-Gröbner basis implicitly does arithmetic modulo 10 in order to make one of the polynomials monic. The lemma from Becker and Weispfenning then states that the leading term of any polynomial in the ideal is divisible by either x^2 or 10. If we have a Coppersmith ideal of the form $\langle N, f \rangle$ for monic quadratic f and known modulus N, this aligns with common choices of shift polynomials $\{N, Nx, f, fx\}$. However, the power of D-Gröbner basis to find shift polynomials is much greater, as is demonstrated by the following algorithm.

4.2 Finding shift polynomials based on monomials

Algorithm 1: Finding optimal shift polynomials

Input : Ideal J, monomial set $\mathcal{M}$
Output: Shift polynomials $\mathcal{S}$ with support in $\mathcal{M}$

Set monomial order $<$ to any weight order
$G \leftarrow$ D-Gröbner basis of $(J, <)$
$\overline{\mathcal{M}} \leftarrow \{m \mid m \leq \max_< \mathcal{M}\}$
$\overline{\mathcal{S}} \leftarrow \{\}$
for $m \in \overline{\mathcal{M}}$ **do**
    $T \leftarrow \{g \in G \mid \mathrm{LM}(g) \text{ divides } m\}$
    if $T \neq \emptyset$ **then**
        $g \leftarrow \mathrm{argmin}_{g \in T} |\mathrm{LC}(g)|$
        $h \leftarrow \frac{m}{\mathrm{LM}(g)} g$
        $\overline{\mathcal{S}} \leftarrow \overline{\mathcal{S}} \cup \{h\}$
$\mathcal{S} \leftarrow$ basis of the subspace of $\mathrm{span}_{\mathbb{Z}}(\overline{\mathcal{S}})$ where all coefficients of $m \notin \mathcal{M}$ are 0
return $\mathcal{S}$

Our method for finding the optimal shift polynomials within an ideal is given in Algorithm 1. At a high level, the algorithm iterates over a set of monomials, and for each monomial m in the set, it finds a polynomial h in ideal J with $\mathrm{LM}(h) = m$. The divisibility property of D-Gröbner bases in Lemma 2 guarantees that we can find an h if one exists. Because $\mathrm{LC}(h)$ contributes directly to the lattice determinant, we choose the candidate h with the smallest leading coefficient.

We must be careful that every polynomial $f \in \mathcal{S}$ has *support* in $\mathcal{M}$, or that the set of monomials that appear in f is a subset of $\mathcal{M}$. To do this, we construct a superset $\overline{\mathcal{M}} \supset \mathcal{M}$ with the property that $\mathrm{LM}(h) \in \overline{\mathcal{M}}$ implies h has support in $\overline{\mathcal{M}}$. This means $\overline{\mathcal{S}}$ has support in $\overline{\mathcal{M}}$, and the linear algebra operations in

line 11 guarantee that $\mathcal{S}$ has support in $\mathcal{M}$. Note that for any weight order, $\overline{\mathcal{M}}$ is finite, so the algorithm terminates.

In many cases in practice, we can use $\overline{\mathcal{M}} = \mathcal{M}$ while still ensuring $\overline{\mathcal{S}}$ has support in $\overline{\mathcal{M}}$. This allows us to skip the linear algebra step, because $\mathcal{S} = \overline{\mathcal{S}}$.

4.3 Optimality of the algorithm

We claim that Algorithm 1 is optimal for a given choice of monomials, but we must define our notion of optimality. In essence, if Coppersmith's method succeeds for a non-optimal set of shift polynomials, it should succeed for the optimal set. We use a combination of properties of both lattices and D-Gröbner bases to prove that the optimal set of shift polynomials for a given set of monomials is unique (up to unimodular transformations) and is found by our algorithm.

For a particular ideal J and choice of monomials $\mathcal{M}$, a vector in the dual lattice is the scaled coefficient vector of a polynomial $f \in J$ where f has support in $\mathcal{M}$. There is a natural embedding $\varphi : J \to \mathbb{Z}^{|\mathcal{M}|}$ that converts shift polynomials to dual lattice vectors. Furthermore, φ is additively homomorphic: $\varphi(f+g) = \varphi(f)+\varphi(g)$, so adding polynomials in the ideal corresponds to adding vectors in a lattice. When the set of shift polynomials $\mathcal{S}$ has support in $\mathcal{M}$, the dual lattice $\Lambda_{\mathcal{S}}$ is the span of vectors $\{\varphi(f) \mid f \in \mathcal{S}\}$.

Informally, we define the *optimal dual lattice* for J and $\mathcal{M}$ as the dual lattice that contains $\varphi(f)$ for every $f \in J$ with support in $\mathcal{M}$. Formally, the proofs become more direct if we define it in terms of sublattices rather than vectors, although the definitions are equivalent. For shift polynomial sets $\mathcal{S}_1$ and $\mathcal{S}_2$,

$$\Lambda_{\mathcal{S}_1 \cup \mathcal{S}_2} = \operatorname{span}\left(\{\varphi(f_1) \mid f_1 \in \mathcal{S}_1\} \cup \{\varphi(f_2) \mid f_2 \in \mathcal{S}_2\}\right) = \Lambda_{\mathcal{S}_1} + \Lambda_{\mathcal{S}_2},$$

so the union of these sets corresponds to the sum of dual lattices (sometimes called the lattice union), and $\Lambda_{\mathcal{S}_1}$ is a sublattice of $\Lambda_{\mathcal{S}_1 \cup \mathcal{S}_2}$.

Definition 2 (Optimal Dual Lattice). *Let $J \subset \mathbb{Z}[\boldsymbol{x}]$ be an ideal, and $\mathcal{M}$ a set of monomials. Any subset $\mathcal{S} \subset J$ of shift polynomials with support in $\mathcal{M}$ defines a dual lattice $\Lambda_{\mathcal{S}}$. The optimal dual lattice Λ^* for $\mathcal{M}$ has the property that $\Lambda^* \supset \Lambda_{\mathcal{S}}$ over all choices of $\mathcal{S}$.*

Lemma 3. *The optimal dual lattice is unique, if it exists.*

Proof. Consider two optimal lattices $\Lambda_{\mathcal{S}_1}$ and $\Lambda_{\mathcal{S}_2}$. $\mathcal{S}_1 \cup \mathcal{S}_2 \subset J$ has support in $\mathcal{M}$, so $\Lambda_{\mathcal{S}_1} \subset \Lambda_{\mathcal{S}_1 \cup \mathcal{S}_2} \subset \Lambda_{\mathcal{S}_1}$ by lattice summation and optimality of $\Lambda_{\mathcal{S}_1}$. Thus $\Lambda_{\mathcal{S}_1} = \Lambda_{\mathcal{S}_1 \cup \mathcal{S}_2}$. The same is true of $\Lambda_{\mathcal{S}_2}$, so therefore $\Lambda_{\mathcal{S}_1} = \Lambda_{\mathcal{S}_2}$.

This definition is also useful for bounding the successive minima of the optimal lattice, which bounds the lengths of vectors found by lattice reduction.

Lemma 4. *For all $\mathcal{S}$ with support in $\mathcal{M}$, the optimal dual lattice Λ^* satisfies*

$$\lambda_i(\Lambda^*) \leq \lambda_i(\Lambda_{\mathcal{S}}) \text{ for } 1 \leq i \leq \operatorname{rank}(\Lambda_{\mathcal{S}}).$$

Proof. Since $\Lambda_{\mathcal{S}} \subset \Lambda^*$, any ball that contains i linearly independent vectors in $\Lambda_{\mathcal{S}}$ also contains the same in Λ^*.

For full-rank dual lattices, as are typically considered in modular Coppersmith problems, this means that the strongest bounds on reduced vector lengths are obtained by considering the optimal lattice. In other words, if vectors are short enough to be found in any dual lattice, they are short enough to be found in the optimal lattice. Finally, we arrive at the main result of this section.

Theorem 1. *Let $\mathcal{S}$ be the shift polynomials returned by Algorithm 1 for ideal J and monomial set $\mathcal{M}$. Then $\{\varphi(f) \mid f \in \mathcal{S}\}$ is a basis for the optimal lattice.*

Proof. The elements of $\mathcal{S}$ are $\mathbb{Z}$-linearly independent, so if $\Lambda_{\mathcal{S}}$ is optimal, then $\mathcal{S}$ defines a basis. It suffices to show that for all possible shift polynomial sets $\mathcal{S}'$, $\Lambda_{\mathcal{S}'} \subset \Lambda_{\mathcal{S}}$. Let $\boldsymbol{v} \in \Lambda_{\mathcal{S}'}$ be a vector, and let $f = \varphi^{-1}(\boldsymbol{v}) \in J$ be the polynomial whose embedding into the dual lattice is $\boldsymbol{v}$. By the homomorphic property of φ, if f is in the integer linear span of $\mathcal{S}$, then $\boldsymbol{v} \in \Lambda_{\mathcal{S}}$, proving $\Lambda_{\mathcal{S}'} \subset \Lambda_{\mathcal{S}}$.

f has support in $\mathcal{M}$, so $f \in \operatorname{span}_{\mathbb{Z}}(\mathcal{S})$ if $f \in \operatorname{span}_{\mathbb{Z}}(\overline{\mathcal{S}})$. We will iteratively subtract integer multiples of elements of $\overline{\mathcal{S}}$ until $f = 0$. Since $\overline{\mathcal{S}} \subset J$, $f \in J$ after each subtraction. First, if $f = 0$, then we are done. If $f \neq 0$, then let $m = \operatorname{LM}(f)$ and let T, g, h be the values in Algorithm 1. T is nonempty by Lemma 2.

Since $\operatorname{LM}(h) = m$ and $h \in \overline{\mathcal{S}}$, subtract an integer multiple of h from f so the coefficient of m in f is in $\{0, 1, \ldots, |\operatorname{LC}(h)| - 1\}$. Note that $\operatorname{LC}(h) = \operatorname{LC}(g)$. Assume that that this coefficient is nonzero. Since $f \in J$, Lemma 2 shows there exists $\hat{g} \in G$ where $\operatorname{LT}(\hat{g}) | \operatorname{LT}(f)$. $\operatorname{LM}(\hat{g}) | \operatorname{LM}(f)$ and $\operatorname{LM}(f) = m$, so $\hat{g} \in T$, but $\operatorname{LC}(\hat{g}) | \operatorname{LC}(f) \Rightarrow |\operatorname{LC}(\hat{g})| \leq |\operatorname{LC}(g)| - 1$, contradicting the minimality of $|\operatorname{LC}(g)|$ in T. Thus the coefficient of m in f is zero, and we have eliminated the leading term. Repeat until $f = 0$.

4.4 Benefits of Our Approach

We compare our strategy to that of Meers and Nowakowski [33]. To solve the Commutative Isogeny Hidden Number Problem for CSURF key exchange, they consider a system of two polynomials modulo prime N with known $\boldsymbol{c}$.

$$f = (c_1 + x_1)^2 + 12(c_1 + x_1) - 4(c_1 + x_1)(c_2 + x_2)^2 - 8(c_2 + x_2)^2 + 36$$
$$g = (c_3 + x_3)^2 + 12(c_3 + x_3) - 4(c_3 + x_3)(c_1 + x_1)^2 - 8(c_1 + x_1)^2 + 36$$

Meers and Nowakoswki's strategy for multiplicity 2 first builds a set $\mathcal{M}$ of 33 monomials, then it finds a set of "optimal" shift polynomials that are products of f and g. However, their algorithm is sensitive to the choice of monomial order, and the resulting lattice determinant can vary. The lexicographic order $x_3 < x_2 < x_1$, lexicographic order $x_2 < x_1 < x_3$, and degree lexicographic order results in determinants proportional to N^{54}, N^{52}, and N^{53}.

Our definition of optimality is independent of monomial order. With the same 33 monomials, Algorithm 1 finds that the optimal lattice has a determinant proportional to N^{51}. This improvement is possible because the Gröbner basis

calculation finds nontrivial polynomial combinations of f and g. For example, our method finds the nontrivial shift polynomial with leading term $Nx_2^2x_3^3$:

$$(-x_1x_3 + (-c_3 - 2)x_1 + (-c_1 + 2)x_3)Nf + (x_2^2 - 4^{-1}x_1 + 2c_2x_2)Ng.$$

The ability to find these nontrivial shift polynomials explains how our approach minimizes the determinant and maximizes performance.

4.5 Provable Monomial Selection Strategy

While it is difficult to know which monomials to include so that the resulting optimal lattice leads to the best performing attack, an easy lemma tells us which monomials to exclude. This lemma can be applied for modular Coppersmith problems where an upper bound P on modulus p is known. This includes problems where p is known or a multiple N of p is known, but it also includes problems like the General Approximate Common Divisors problem [12] where no multiple is known.

Lemma 5. *We are given bounds $\boldsymbol{X}$, ideal J in which all polynomials share a root modulo p, and upper bound $P \geq p$. Let $\mathcal{M}$ be any monomial set. Define*

$$\mathcal{M}_{opt} = \{m \mid m(\boldsymbol{X}) < P\}.$$

Let Λ be the optimal dual lattice of $\mathcal{M}$ and J. If vector $\boldsymbol{v} \in \Lambda$ satisfies the bound of Håstad/Howgrave-Graham, then the polynomial corresponding to $\boldsymbol{v}$ has support in $\mathcal{M} \cap \mathcal{M}_{opt}$.

Proof. Let $g = \varphi^{-1}(\boldsymbol{v})$ be the polynomial corresponding to $\boldsymbol{v}$. Consider a monomial $m \in \mathcal{M} \setminus \mathcal{M}_{\text{opt}}$. The coefficient of m in g is $c_m \in \mathbb{Z}$, so the entry at index m in $\boldsymbol{v}$ is an integer multiple c_m of $m(\boldsymbol{X})$. If $c_m \neq 0$, then $\|\boldsymbol{v}\| \geq m(\boldsymbol{X}) \geq P \geq p$. This contradicts the assertion that $\boldsymbol{v}$ satisfies the HHG bound, since $\|\boldsymbol{v}\| < p/\sqrt{|\mathcal{M}|} \leq p$. Therefore $c_m = 0$, so g has support in $\mathcal{M} \cap \mathcal{M}_{\text{opt}}$.

In other words, it is unhelpful to consider any monomials not in $\mathcal{M}_{\text{opt}}$, because the coefficients will always be zero in the recovered polynomials. Additionally, if a short vector satisfying the HHG bound exists in the optimal lattice for $\mathcal{M}$, a vector of the exact same length (corresponding to the same polynomial) must exist in the optimal lattice for $\mathcal{M}_{\text{opt}}$.

This gives us the provable strategy; for an ideal J with modulus p, we run Algorithm 1 with $\mathcal{M}_{\text{opt}}$ to find the shift polynomials for the optimal lattice. While this strategy is fully automatic and has provable guarantees, $\mathcal{M}_{\text{opt}}$ may be quite large, meaning lattice reduction is computationally expensive and the exponential term in lattice reduction vector bounds is large. To improve the practical performance of our method further, we will require a heuristic monomial selection strategy.

5 Heuristic Monomial Selection

Section 4 describes a provable strategy for taking an ideal J and selecting monomials $\mathcal{M}_{\text{opt}}$ and shift polynomials that are guaranteed to produce a dual Coppersmith lattice with useful properties. If a vector in any other dual lattice is short enough to satisfy the HHG bound, an equally short vector exists in our provable lattice.

However, $|\mathcal{M}_{\text{opt}}|$ is large, and the lattice has large rank, meaning lattice reduction is expensive and may not be guaranteed to find the short vectors that exist. If we can directly construct a basis for a low-rank sublattice which still contains suitably short vectors, then lattice reduction becomes practically faster and yields better bounds. This section describes a heuristic strategy for finding shift polynomials which lead to an improved sublattice of the optimal lattice.

At a high level, we bound the length of sublattice vectors using the determinant of the sublattice. It's easy to compute the determinant when the lattice is full-rank and the basis is triangular, so we assume that Algorithm 1 returns a basis $\mathcal{S}$ that is triangularized according to some monomial order $<$, noting that triangularization of a basis is computationally efficient. The requirement that the lattice basis is full-rank and triangular is captured by the notion of $(\mathcal{M}, <)$-*suitability*.

Definition 3 ([33]). *Given monomial set $\mathcal{M}$ and monomial ordering $<$, a set of shift polynomials $\mathcal{S}$ is $(\mathcal{M}, <)$-**suitable** if every $f \in \mathcal{S}$ has support in $\mathcal{M}$, and for each $m \in \mathcal{M}$, there is a unique $f \in \mathcal{S}$ with $\text{LM}(f) = m$.*

For a $(\mathcal{M}, <)$-suitable sets $\mathcal{S}$ of shift polynomials, the dual lattice is full-rank, has dimension $|\mathcal{S}|$, and has determinant $\det \Lambda_{\mathcal{S}} = \prod_{f \in \mathcal{S}} \text{LT}(f)(\boldsymbol{X})$. Since our focus is modular Coppersmith problems where a multiple N of modulus p is known, ideal J contains the constant polynomial N. When J has this property, Algorithm 1 returns a $(\mathcal{M}, <)$-suitable set $\mathcal{S}$. This is because Lemma 2 guarantees that the D-Gröbner basis of J includes a polynomial g with $\text{LM}(g) = 1$, so T is always nonempty in line 7. The span of $\overline{\mathcal{S}}$ has rank $|\overline{\mathcal{M}}|$, meaning the span of $\mathcal{S}$ has rank $|\mathcal{M}|$ and the triangularized basis is $(\mathcal{M}, <)$-suitable.

5.1 Sublattice Structure

The sublattice structure of Coppersmith lattices addresses one of the major open questions of prior research: Coppersmith's method often far outperforms expectations, finding lattice vectors significantly shorter than predicted by the determinant bound $\det(\Lambda)^{1/\operatorname{rank}(\Lambda)}$. These unexpectedly short vectors belong to *dense* sublattices, or sublattices $\Lambda_{\text{sub}} \subset \Lambda$ where

$$\det(\Lambda_{\text{sub}})^{1/\operatorname{rank}(\Lambda_{\text{sub}})} \ll \det(\Lambda)^{1/\operatorname{rank}(\Lambda)},$$

so understanding the sublattice structure has two benefits. First, it helps us close the gap between theoretical performance of Coppersmith's method and experimental results. Second, if we directly construct a sufficiently dense sublattice of

the optimal Coppersmith lattice, the decreased rank leads to faster lattice reduction and improved practical performance. We qualitatively observe four causes of dense sublattices in Coppersmith lattices.

First, many instances of Coppersmith problems assume the existence of a small root. This implies the existence of a small vector in the Coppersmith primal lattice, and duality [35] implies the existence of a dense sublattice of rank $|\mathcal{S}| - 1$ in the Coppersmith dual lattice. This explains the gap between theory and practice in [19]. We probably cannot construct this sublattice directly (since then we could construct the small root directly), but we can intentionally construct instances without small roots, allowing us to study other causes of dense sublattices.

Second, the choice of shift polynomial ideal can explain some of the sublattice structure. If the optimal lattice for ideal J includes a dense sublattice, then the optimal lattice for J^2 amplifies this structure, since $g \in J$ with small coefficients implies $g^2 \in J^2$ with small coefficients. These complex dense sublattices are hard to analyze, but easy to avoid in practice: simply use as small a multiplicity as possible. This sublattice structure explains the gap between theory in practice in [12], particularly the entries in Table 1 which report an LLL factor ≈ 0.5.

Third, the basis of the dual lattice is often sparse, meaning orthogonality in projected sublattices is more common than expected for random lattices. Since the sparsity of shift polynomials is known, it's possible to analyze the nonzero coefficients of shift polynomials to find improved sublattices. This is explored in more detail in Section 5.2.

Fourth, shift polynomials sometimes have small, non-zero coefficients. This leads to near-orthogonality in projected sublattices, which is harder to analyze. Unravelled linearization can help here, and this is explored in Section 5.3.

5.2 Sparse Polynomials and Graph Search

Size reduction of the triangular basis returned by Algorithm 1 often results in sparse shift polynomials. Size reduction ensures that non-leading coefficients in the basis are small relative to leading coefficients, but it is also directly related to the unique normal form of a polynomial with respect to a D-Gröbner basis [3, Theorem 10.23]. Sparse shift polynomials mean that many of the entries in the dual lattice basis are 0, and we can often use this sparsity to find dense sublattices. For example, consider the dual bases for $\mathcal{S} = \{N, Nx, x^2 + a\}$ and $\mathcal{S}_{\text{sub}} = \{N, x^2 + a\}$:

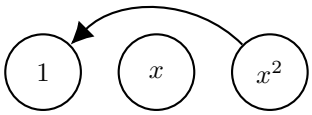
$$B = \begin{pmatrix} N & 0 & 0 \\ 0 & NX & 0 \\ a & 0 & X^2 \end{pmatrix} \qquad B_{\text{sub}} = \begin{pmatrix} N & 0 & 0 \\ a & 0 & X^2 \end{pmatrix}$$

Basis B has determinant N^2X^3 and rank 3. Because the coefficient of x in $\mathcal{S}_{\text{sub}}$ is always zero, observe that the second column of B_{sub} can be eliminated without affecting the lattice vector lengths. This makes it easy to compute that the lattice spanned by B_{sub} has determinant NX^2 and rank 2. Since $(NX^2)^{1/2} < (N^2X^3)^{1/3}$, this is a dense sublattice.

While the sublattice is easy to identify in this toy example, the problem becomes more difficult when $\mathcal{S}$ contains hundreds or thousands of sparse polynomials. We must search over all possible subsets of $\mathcal{S}$, consider subsets that contain all-zero columns in the corresponding basis matrix (so the determinant is easy to compute), and compare the density of the sublattice to the density of the original lattice. The main contribution of this section is a method based on graph optimization algorithms that does this, automatically and efficiently identifying these dense sublattices.

The requirement for all-zero columns is related to shift polynomial suitability. If $\mathcal{S}$ is $(\mathcal{M}, <)$-suitable, and $\mathcal{S}_{\text{sub}} \subset \mathcal{S}$ is $(\mathcal{M}_{\text{sub}}, <)$-suitable, then this means $\mathcal{S}_{\text{sub}}$ has support in $\mathcal{M}_{\text{sub}}$, and the columns corresponding to monomials $\mathcal{M} \setminus \mathcal{M}_{\text{sub}}$ are necessarily all zero. Our algorithm involves a directed graph that encodes information about shift polynomial suitability.

The directed graph represents dependencies between monomials in polynomials in $\mathcal{S}$. Given a $(\mathcal{M}, <)$-suitable set of shift polynomials $\mathcal{S}$, we define a graph $\mathcal{G}$ with vertices $\mathcal{M}$ and directed edges (m_1, m_2) if $m_1 \neq m_2$ and $\exists f \in \mathcal{S}$ with $\text{LM}(f) = m_1$ and the coefficient of m_2 in f is nonzero. Once again, consider the $(\{1, x, x^2\}, <)$-suitable $\mathcal{S} = \{N, Nx, x^2 + a\}$. The corresponding graph follows.



Directed edges denote dependencies for $(\mathcal{M}_{\text{sub}}, <)$-suitable subsets $\mathcal{S}_{\text{sub}} \subset \mathcal{S}$. If $x^2 \in \mathcal{M}_{\text{sub}}$, then $x^2 + a \in \mathcal{S}_{\text{sub}}$, implying $1 \in \mathcal{M}_{\text{sub}}$. Indeed, $\{N, x^2 + a\}$ is a $(\{1, x^2\}, <)$-suitable set. Finding a suitable subset is therefore equivalent to finding a subgraph where there are no edges leading out of the subgraph (so there are no unmet dependencies). This is called a closure.

Definition 4 (Graph Closure). *Let $\mathcal{G} = (V, E)$ be a directed graph. $V' \subset V$ is a closure if there exists no directed edge $(v_1, v_2) \in E$ with $v_1 \in V'$ and $v_2 \in V \setminus V'$.*

Picard [37] studied the problem of finding a closure of maximum total weight in a vertex-weighted directed graph. He proposed an algorithm which efficiently finds the maximal closure by reducing the problem to an equivalent maximal flow problem and solving with the Ford-Fulkerson algorithm. We will use Picard's algorithm as a subroutine to find a $(\mathcal{M}_{\text{sub}}, <)$-suitable proper subset $\mathcal{S}_{\text{sub}} \subsetneq \mathcal{S}$ with better determinant bounds if one exists. This process is documented in Algorithm 2. Intuitively, we want to find subsets with small determinant, so if we weight vertices by $-\log \text{LT}(f)(\boldsymbol{X})$, then a closure which *maximizes the sum* of the logarithmic weights is a subset which *minimizes the product* $\prod_{f \in \mathcal{S}_{\text{sub}}} \text{LT}(f)(\boldsymbol{X})$. In our algorithm, we use similar weights,[4] call Picard's algorithm to find a closure, and check if the closure is a proper subset.

[4] They are different because we want small $\det(\Lambda_{\mathcal{S}_{\text{sub}}})^{1/|\mathcal{S}_{\text{sub}}|}$, not small $\det(\Lambda_{\mathcal{S}_{\text{sub}}})$

Algorithm 2: Finding a suitable subset of shift polynomials

Input : Bounds $\boldsymbol{X}$, monomial set $\mathcal{M}$ and $(\mathcal{M}, <)$-suitable $\mathcal{S}$
Output: $\mathcal{M}_{\text{sub}} \subsetneq \mathcal{M}$ and $(\mathcal{M}_{\text{sub}}, <)$-suitable $\mathcal{S}_{\text{sub}} \subsetneq \mathcal{S}$ with better determinant bounds if one exists, else $\perp$

$E \leftarrow \{(m_1, m_2) \mid \exists f \in \mathcal{S}$ with $\text{LM}(f) = m_1$ and $m_2 \neq m_1$ is a monomial in $f\}$
Construct directed graph $\mathcal{G} \leftarrow (\mathcal{M}, E)$
for $f_m \in \mathcal{S}$ *with* $\text{LM}(f_m) = m$ **do**
  `// Set weight of vertex` $m \in \mathcal{M}$
  $w_m \leftarrow -\log_2(\text{LT}(f_m)(\boldsymbol{X})) + \frac{1}{|\mathcal{S}|}\sum_{g\in\mathcal{S}} \log_2(\text{LT}(g)(\boldsymbol{X}))$
$\mathcal{M}_{\text{sub}} \leftarrow$ maximal closure of $\mathcal{G}$ with weights $\{w_m\}_{m\in\mathcal{M}}$
if $\sum_{m\in\mathcal{M}_{sub}} w_m = 0$ **then**
  return $\perp$
else
  $\mathcal{S}_{\text{sub}} \leftarrow \{f \in \mathcal{S} \mid \text{LM}(f) \in \mathcal{M}_{\text{sub}}\}$
  return $\mathcal{M}_{\text{sub}}, \mathcal{S}_{\text{sub}}$

Theorem 2. *Algorithm 2 is correct. On input $\boldsymbol{X}, \mathcal{M}, \mathcal{S}$, it returns a proper, nonempty subset $\mathcal{M}_{sub}$ and corresponding $(\mathcal{M}_{sub}, <)$-suitable $\mathcal{S}_{sub} \subsetneq \mathcal{S}$ where*

$$\det(\Lambda_{\mathcal{S}_{sub}})^{1/|\mathcal{S}_{sub}|} < \det(\Lambda_{\mathcal{S}})^{1/|\mathcal{S}|}$$

if such a proper subset exists, otherwise it returns $\perp$.

Proof. Observe that any closure of $\mathcal{G}$ corresponds to a suitable subset of $\mathcal{S}$. Consider closures $\mathcal{M}' = \emptyset$ and $\mathcal{M}' = \mathcal{M}$; these closures have weight $\sum_{m\in\mathcal{M}'} w_m = 0$. Note that for closure $\mathcal{M}'$ and corresponding $\mathcal{S}' \subset \mathcal{S}$,

$$\det(\Lambda_{\mathcal{S}'})^{1/|\mathcal{S}'|} < \det(\Lambda_{\mathcal{S}})^{1/|\mathcal{S}|} \Leftrightarrow \prod_{f\in\mathcal{S}'} \text{LT}(f)(\boldsymbol{X}) < \prod_{f\in\mathcal{S}} \text{LT}(f)(\boldsymbol{X})^{|\mathcal{S}'|/|\mathcal{S}|}$$

$$\Leftrightarrow \sum_{f\in\mathcal{S}'} \log_2 \text{LT}(f)(\boldsymbol{X}) < \frac{|\mathcal{S}'|}{|\mathcal{S}|}\sum_{g\in\mathcal{S}} \log_2(\text{LT}(g)(\boldsymbol{X})) \Leftrightarrow 0 < \sum_{m\in\mathcal{M}'} w_m.$$

That is, if a proper nonempty subset $\mathcal{M}'$ exists with improved determinant bounds, then the maximal closure has positive weight. Picard's algorithm finds the maximal closure, which corresponds to a proper nonempty subset $\mathcal{M}_{\text{sub}}$. If no such proper nonempty subset exists, then the maximal closure has total weight 0, and Algorithm 2 returns $\perp$.

To obtain $\mathcal{M}_{\text{heur}}$, we begin with $\mathcal{M}_{\text{opt}}$. We find that inclusion of $f \in \hat{J}_\infty$ in $\mathcal{S}$ leads to spurious short vectors that fail to satisfy Heuristic 1, so we set $\mathcal{M}' = \mathcal{M}_{\text{opt}} \setminus \text{LM}(\hat{J}_\infty)$, which is easily computed from the Gröbner basis of $\hat{J}_\infty$. Next, we iteratively apply Algorithm 2 to $\mathcal{M}'$ until no more proper subsets are found.[5] This final set is $\mathcal{M}_{\text{heur}}$.

[5] Incidentally, this iterated process is guaranteed to minimize $\det(\Lambda_{\mathcal{S}_{\text{heur}}})^{1/|\mathcal{S}_{\text{heur}}|}$ over possible closures, which we prove in Appendix B.

5.3 Small Coefficients and Unravelled Linearization

In some cases, the shift polynomials have small coefficients. For example, Boneh and Durfee's approach to the small private exponent RSA problem [7] can lead to $\mathcal{S} = \{N, Nx_1, x_1x_2 + ax_1 + 1\}$ and basis

$$B = \begin{pmatrix} N & 0 & 0 \\ 0 & NX_1 & 0 \\ 1 & aX_1 & X_1X_2 \end{pmatrix}.$$

Because the constant coefficient 1 in $x_1x_2 + ax_1 + 1$ is small relative to N, the second and third row are very nearly orthogonal to the first row. The rank-2 sublattice they span has determinant close to $NX_1^2X_2$, but Algorithm 2 fails to find this sublattice because $\{Nx_1, x_1x_2 + ax_1 + 1\}$ is not suitable.

To make the determinant easier to compute, Herrmann and May introduced the variable $u = x_1x_2 + 1$ with bound $U = X_1X_2 + 1$ [20]. The lattice bases for $\mathcal{S}' = \{N, Nx_1, u + ax_1\}$ and $\mathcal{S}'_{\text{sub}} = \{Nx_1, u + ax_1\}$ are then

$$B' = \begin{pmatrix} N & 0 & 0 \\ 0 & NX_1 & 0 \\ 0 & aX_1 & U \end{pmatrix} \qquad B'_{\text{sub}} = \begin{pmatrix} 0 & NX_1 & 0 \\ 0 & aX_1 & U \end{pmatrix}.$$

With unravelled linearization, the small, nonzero coefficients vanish, $\mathcal{S}'_{\text{sub}}$ is $(\{x_1, u\}, <)$-suitable, and Algorithm 2 can find the sublattice with determinant $NX_1U \approx NX_1^2X_2$. In general, when Algorithm 2 fails due to small coefficients, it is often helpful to apply unravelled linearization.

6 Asymptotically Fast Use of Precomputation

The shift polynomial strategies in Sections 4 and 5 are powerful and effective, but the cost of Gröbner basis computation becomes increasingly expensive as the multiplicity grows. In addition, it is challenging to analyze the asymptotic behavior of our approaches. To remedy this, we propose a third strategy based on symbolic precomputation. In the precomputation phase, we use a symbolic representation of the input polynomials to calculate symbolic shift polynomials. Once the actual coefficients of the input polynomials are known, the shift polynomials can be computed quickly and explicitly. We also describe an approach to extend these shift polynomials to higher multiplicities. Our approach appears to satisfy a modified heuristic assumption by Meers and Nowakowski [33], enabling us to automatically determine asymptotic bounds for a precomputed set of shift polynomials.

There are trade-offs for this approach. Unlike our previous shift polynomial strategies, this method is not fully automated and requires guessing a good monomial set before computing the symbolic shift polynomials. It loses the guarantee that the dual lattice is optimal, and it may be less effective at recovering roots for a fixed multiplicity. However, it is fast in practice, and it achieves the same asymptotic bounds and heuristic polynomial running time guarantees that are found in prior work without the need for tedious manual calculations.

6.1 Symbolic Representation of Shift Polynomials

In many cases, the algorithms of Sections 3, 4, and 5 can be performed symbolically. Specifically, we consider the case where there is a single modulus p and a multiple N of the modulus is known. We use variables $\boldsymbol{c}$ to represent the coefficients which are known during an attack, but unknown in advance, and work in the fraction field of the polynomial ring $\mathbb{Q}[\boldsymbol{c}]$, which we denote by $\mathbb{K}_{\boldsymbol{c}}$. We specify input relations by the polynomial ring $\mathbb{K}_{\boldsymbol{c}}[N, \boldsymbol{x}]$. We build ideals, calculate Gröbner bases, and run Algorithm 1 in this polynomial ring.

For example, consider the problem of factoring RSA modulus $N = pq$ when the least significant bits of p are known [34]. N is a known multiple of p, and we may use relation $f(N, x) = c_1 x + c_2$ where x represents the most significant bits, c_1 is a power of two, and c_2 represents the known least significant bits. Members of the ideal $J = \langle N, f \rangle \subset \mathbb{K}_{\boldsymbol{c}}[N, x]$ share a root modulo p, and the Gröbner basis of J^2 is

$$\left\{ N^2, \qquad Nx + \frac{c_2}{c_1} N, \qquad x^2 + \frac{2c_2}{c_1} x + \frac{c_2^2}{c_1^2} \right\}.$$

Therefore, by specifying the symbolic input relations, a desired multiplicity k_{pre}, and a set of monomials, we can compute $\hat{J}_p^{k_{\text{pre}}} + \hat{J}_\infty$ and use Algorithm 1 to return a symbolic representation of the shift polynomials $\mathcal{S}_1 \subset \hat{J}_p^{k_{\text{pre}}} + \hat{J}_\infty$. If $\hat{J}_\infty$ is nontrivial, we also compute a symbolic representation of its Gröbner basis. During an attack, once the values of coefficients $\boldsymbol{c}$ are known, they may be substituted into the precomputed $\mathcal{S}_1$. Division in fraction field $\mathbb{K}_{\boldsymbol{c}}$ is replaced by inversion modulo $N^{k_{\text{pre}}}$, so this substitution requires that the denominators in $\mathcal{S}$ are coprime to N. In our example, c_1 is a power of two, which is coprime to the RSA modulus N.

6.2 Extending to Higher Multiplicities

After substituting in the known coefficients, we have shift polynomials $\mathcal{S}_1$ which share a root modulo $p^{k_{\text{pre}}}$ and belong to an ideal $\hat{J}_p^{k_{\text{pre}}} + \hat{J}_\infty \subset \mathbb{Z}[x]$. We use $\mathcal{S}_1$ to compute shift polynomials with higher multiplicities to avoid calculating the Gröbner basis of $\hat{J}_p^{k k_{\text{pre}}} + \hat{J}_\infty$ during the attack itself.

Given a desired multiplicity k and parameter $\boldsymbol{t} \in \mathbb{Z}_{\geq 0}^{\ell}$, we have a three step process to compute shift polynomials $\mathcal{S}_{k,\boldsymbol{t},\text{ul}} \subset \hat{J}_p^{k k_{\text{pre}}} + \hat{J}_\infty$. We rely on a filtration operation Φ, where if multiple shift polynomials share a leading monomial, we keep the one with the smallest leading coefficient:

$$\Phi(\mathcal{S}) = \left\{ \text{argmin}_{f \in S, \text{LM}(f) = m} |\,\text{LC}(f)| \mid m \in \text{LM}(S) \right\}$$

Computing polynomials in $(\hat{J}_p^{k_{\text{pre}}} + \hat{J}_\infty)^k$. We recursively construct

$$\mathcal{S}_k = \Phi\left(\{ f f' \mid (f, f') \in \mathcal{S}_1 \times \mathcal{S}_{k-1} \} \right).$$

This set grows similarly to the monomial sets from Jochemsz and May [22] and Meers and Nowakowski [33], which consider terms in f^k and $\prod_i f_i^k$ respectively.

Computing x-shifts in $(\hat{J}_p^{k\mathbf{pre}} + \hat{J}_\infty)^k$. In some cases, the bound X_i is small, so it is beneficial to include extra monomials involving x_i. Using $\boldsymbol{t}$, we compute

$$\mathcal{S}_{k,\boldsymbol{t}} = \Phi\left(\mathcal{S}_k \cup \left\{\prod_{i=1}^{\ell} x_i^{e_i} f \mid f \in \mathcal{S}_k, 0 \le e_i \le t_i \forall i \in \{1,\ldots,\ell\}\right\}\right).$$

This is like the extended strategy in [22] and the y-shifts of Boneh and Durfee [7].

Unravelling into $\hat{J}_p^{kk\mathbf{pre}} + \hat{J}_\infty$. As described in Section 5.2, it is empirically helpful to exclude monomials in $\mathrm{LM}(\hat{J}_\infty)$ when using unravelled linearization.

$$\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}} = \left\{\mathrm{normal\_form}_{\mathrm{GB}(\hat{J}_\infty)}(f) \mid f \in \mathcal{S}_{k,\boldsymbol{t}}, \mathrm{LM}(f) \notin \mathrm{LM}(\hat{J}_\infty)\right\}.$$

Assuming the polynomials in $\mathrm{GB}(\hat{J}_\infty)$ are monic, the normal form [3] of f ensures that no monomial in $\mathrm{LM}(\hat{J}_\infty)$ appears in $\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}}$.

For this strategy to be efficient, $|\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}}|$ cannot grow too quickly, but a naïve bound on $|\mathcal{S}_k| \le |\mathcal{S}_1|^k$ is exponential in k. One goal of this section is to show that for the proper choice of $\mathcal{M}$, the bound is actually polynomial in k.

6.3 Specifying Monomials for Precomputation

In order to use Algorithm 1 to find a symbolic representation of shift polynomials $\mathcal{S}_1$, we must specify a set of monomials $\mathcal{M}_1$. Our choice of representation is related to the theory of *Newton polytopes*, which have previously been used to analyze the asymptotic behavior of Coppersmith's method. To our knowledge, they were first used by Blömer and May in 2005 [6] to analyze bivariate integer Coppersmith problems.

Concurrent work by Feng, Luo, Chen, Nitaj, and Pan also uses Newton polytopes to compute asymptotic bounds [17]. They arrive at provable bounds without relying on heuristic assumptions like this work and [33] do. However, the Newton polytope in their work is determined by the input polynomials, and they analyze the asymptotics of that polytope. In contrast, the polytope in our work is specified independently.

To define the Newton polytope, note that monomials $\prod_{i=1}^{\ell} x_i^{e_i}$ naturally biject with nonnegative integer points $(e_1,\ldots,e_\ell)$ in ℓ-dimensional space. The Newton polytope $\mathcal{P}_f$ of a polynomial f is the convex hull of its monomials. By convexity, if $\mathcal{P}_f$ and $\mathcal{P}_g$ are the Newton polytopes of polynomials f and g, then the Minkowski sum $\mathcal{P}_f + \mathcal{P}_g$ (where each point is the sum of a point in $\mathcal{P}_f$ and in $\mathcal{P}_g$) is the Newton polytope of product fg (where each monomial is the product of a monomial in f and in g).

We specify the precomputed monomial set $\mathcal{M}_1$ using convex polytopes. Given a set $\mathcal{M}_{\mathrm{vert}}$ of monomials representing the vertices of $\mathcal{P}_1 = \mathrm{ConvexHull}(\mathcal{M}_{\mathrm{vert}})$,

$$\mathcal{M}_1 = \left\{\prod_{i=1}^{\ell} x_i^{e_i} \middle| \boldsymbol{e} \text{ is an integer point in } \mathcal{P}_1\right\}.$$

$\mathcal{S}_1$ is computed using the symbolic ideal and Algorithm 1. Since a multiple of the modulus is known, $\mathcal{S}_1$ is $(\mathcal{M}_1, <)$-suitable, and there is a one-to-one correspondence with $\mathcal{M}_1$. We geometrically represent $\mathcal{S}_1$ by the polytope $\mathcal{P}_1$, and the sets $\mathcal{S}_k, \mathcal{S}_{k,\boldsymbol{t}}, \mathcal{S}_{k,\boldsymbol{t},\text{ul}}$ have geometric interpretations $\mathcal{P}_k, \mathcal{P}_{k,\boldsymbol{t}}, \mathcal{P}_{k,\boldsymbol{t},\text{ul}}$ as well.

- If $\mathcal{P}_1$ and $\mathcal{P}_{k-1}$ are polytopes corresponding to $\mathcal{S}_1$ and $\mathcal{S}_{k-1}$, then $\mathcal{P}_k = \mathcal{P}_1 + \mathcal{P}_{k-1}$ corresponds to $\mathcal{S}_k$. This follows from the properties of Newton polytopes and the convexity of $\mathcal{P}_1$. By induction, $\mathcal{P}_k = k\mathcal{P}_1$, a scaled version (dilation) of $\mathcal{P}_1$.
- If $\mathcal{P}_k$ corresponds to $\mathcal{S}_k$, then $\mathcal{P}_{k,\boldsymbol{t}}$ is the union of translations of $\mathcal{P}_k$. The maximum translation in each dimension is t_i.
- $\text{LM}(\hat{J}_\infty)$ corresponds to a union of cones, so $\mathcal{P}_{k,\boldsymbol{t},\text{ul}}$ is the (non-convex) polytope of $\mathcal{P}_{k,\boldsymbol{t}}$ minus the cones.

These polytopes are depicted in Figure 1.

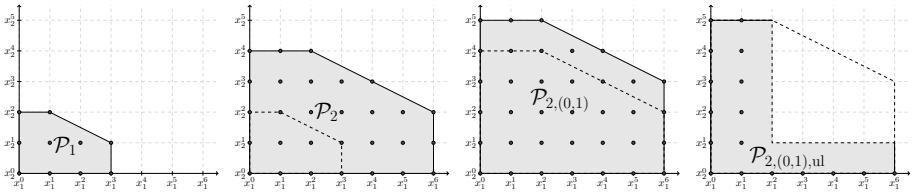


Fig. 1: **Polytopes corresponding to our shift polynomial sets.** Polytope $\mathcal{P}_1$ is defined by vertices $\mathcal{M}_{\text{vert}} = \{1, x_1^3, x_1^3x_2, x_1x_2^2, x_2^2\}$, and $\hat{J}_\infty = \langle x_1^2x_2 - x_1 \rangle$. These correspond to the monomials that appear in $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_{2,(0,1)}$, and $\mathcal{S}_{2,(0,1),\text{ul}}$. Polytope $\mathcal{P}_2$ is $\mathcal{P}_1$ scaled by 2. Polytope $\mathcal{P}_{2,(0,1)}$ is $\mathcal{P}_2$ along with monomials from x_2 shifts. $\mathcal{P}_{2,(0,1),\text{ul}}$ eliminates all monomials that are multiples of $x_1^2x_2$.

6.4 Computing lattice properties with polytopes

The dimension of the Coppersmith lattice for a $(\mathcal{M}, <)$-suitable set $\mathcal{S}$ is given by $|\mathcal{M}|$, which happens to be the number of integer points in the polytope:

$$\dim \Lambda_{\mathcal{S}} = \#(\mathcal{P} \cap \mathbb{Z}^\ell).$$

Similarly, the contributions of $\log X_i$ to the log-determinant can also be expressed using the polytope:

$$\log \prod_{m \in \mathcal{M}} m(\boldsymbol{X}) = \sum_{\boldsymbol{x^e} \in \mathcal{M}} \sum_{i=1}^{\ell} e_i \log X_i = \sum_{i=1}^{\ell} \left(\sum_{\boldsymbol{e} \in \mathcal{P}} e_i \right) \log X_i.$$

As in prior work, we introduce functions $s_{\dim}$, $\{s_{x_i}\}_{1 \le i \le \ell}$, and $\{s_{C_j}\}_{C_j \in \text{LC}(\mathcal{S}_1)}$ to represent certain terms in the dimension and log-determinant expressions for

the lattice corresponding to $\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}}$:

$$\dim \Lambda_{\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}}} = s_{\dim}(k, \boldsymbol{t})$$
$$\log\det \Lambda_{\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}}} = \sum_{i=1}^{\ell} s_{x_i}(k, \boldsymbol{t}) \log X_i + \sum_{C_j \in \mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k, \boldsymbol{t}) \log C_j.$$

It is clear that $s_{\dim}$ and s_{x_i} are weighted sums over integer points in polytopes.

This connection to polytopes allows us to tap into the rich field of Ehrhart theory. A foundational result of the field states that the number of integer points in a k-dilation of a polytope is described by a polynomial called the *Ehrhart polynomial*. As a result, we may use Ehrhart theory to bound the complexity of our precomputation strategy. We refer to the introduction of [10] for background on Ehrhart polynomials and the theorems we cite in this proof. The application of Ehrhart theory to Coppersmith's method was independently introduced by Feng at al. in concurrent work [17].

Lemma 6. *Let $\mathcal{P}_1$ and $\mathcal{S}_1$ be a convex polytope and shift polynomial set as previously defined. Then $|\mathcal{S}_k|$ is polynomial in k. Additionally, if $\hat{J}_\infty = \{0\}$ and we fix $\boldsymbol{t} = \mathbf{0}$, then $s_{dim}(k, \mathbf{0})$ and $s_{x_i}(k, \mathbf{0})$ are both polynomial in k with degrees $\dim \mathcal{P}_1$ and $\dim \mathcal{P}_1 + 1$ respectively.*

Proof. Because $\mathcal{P}_k$ is a k-dilation of $\mathcal{P}_1$, $|\mathcal{S}_k| = \#(k\mathcal{P} \cap \mathbb{Z}^\ell)$. Thus $|\mathcal{S}_k|$ is described by the Ehrhart polynomial [15] of $\mathcal{P}_1$, which has degree $\dim \mathcal{P}_1 \leq \ell$ (the dimension of the polytope is unrelated to the dimension of the lattice). Since $\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}} = \mathcal{S}_{k,\boldsymbol{t}} = \mathcal{S}_k$, $s_{\dim}(k, \mathbf{0}) = |\mathcal{S}_k|$ is polynomial in k.

We have $s_{x_i}(k, \mathbf{0}) = \sum_{\boldsymbol{e} \in k\mathcal{P}} e_i$, which is a sum over integer points in a dilated polytope, weighted by a homogeneous polynomial of degree 1. A result of Brion and Vergne [9] proves that s_{x_i} is a polynomial of degree $\dim \mathcal{P} + 1$.

Since $|\mathcal{S}_k|$ is polynomial in k, it's simple to prove that $|\mathcal{S}_{k,\boldsymbol{t}}| \leq (\max \boldsymbol{t})^\ell |\mathcal{S}_k|$ is bounded by a polynomial in k and $\boldsymbol{t}$, and $|\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}}|$ is also asymptotically polynomial. This means that $\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}}$ can be constructed in polynomial time. However, to analyze the asymptotic behavior further, we need to rely on a heuristic assumption. This assumption is essentially the same as that by Meers and Nowakowski [33] and is well supported by experiment.

Heuristic 2 *Let $\mathcal{S}_1$ represent a convex polytope, and let $\mathcal{S}_{k,\boldsymbol{t},ul}$ be as previously defined. Then if $\mathcal{S}_{k,\boldsymbol{t},ul}$ is suitable, the functions s_{dim}, s_{x_i}, and s_{C_j} which define the lattice dimension and determinant are polynomials in both k and $\boldsymbol{t}$ and have maximum total degree $\ell + 1$.*

As an example of these polynomials, consider the polytope in Figure 1. We have that $|\mathcal{S}_{1,(0,0)}| = 10$, $|\mathcal{S}_{2,(0,0)}| = 29$, and $|\mathcal{S}_{2,(0,1)}| = 36$. This is satisfied by

$$|\mathcal{S}_{k,\boldsymbol{t}}| = 5k^2 + 2kt_1 + 3kt_2 + t_1t_2 + 4k + t_1 + t_2 + 1.$$

Similarly, $s_{\dim}(k, t_1, t_2) = |\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}}| = 7k + t_1 + 2t_2 + 1$ agrees with $|\mathcal{S}_{2,(0,1),\mathrm{ul}}| = 17$.

6.5 Asymptotic Behavior

Meers and Nowakowski used their heuristic assumption to analyze asymptotic behavior as $k \to \infty$. We do the same here. The following lemma describes the bounds $\boldsymbol{X}$ for which a Coppersmith problem is solvable, if we can take p to be arbitrarily large. An extended proof of this lemma and an example application are found in Appendix C.

Lemma 7. *Consider a multivariate Coppersmith problem with known (multiple of) modulus p where bound X_i is given by constants a_i and b_i and variable δ:*

$$\log X_i = (a_i + b_i\delta)\log p.$$

Given a precomputed shift polynomial set $\mathcal{S}_1$ with shared root modulo p, let s_{dim}, s_{x_i}, and s_{C_j} represent the functions describing the parameterized lattice dimension and determinant. Assume Heuristics 1 and 2 hold, and assume the values substituted into the precomputation are proportional to p. For sufficiently large p and any $\boldsymbol{\tau} \geq 0$, the Coppersmith problem is solvable for $\delta < \delta^* - \epsilon$ where $\delta^* =$

$$\lim_{k\to\infty} \frac{k s_{dim}(k, k\boldsymbol{\tau}) - \sum_{i=1}^{\ell} a_i s_{x_i}(k, k\boldsymbol{\tau}) - \sum_{C_j \in \mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k, k\boldsymbol{\tau}) \log_p C_j}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, k\boldsymbol{\tau})} \tag{3}$$

In particular, for multiplicity k we require

$$\log p = \omega\left(\frac{s_{dim}(k, k\boldsymbol{\tau})^2}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, k\boldsymbol{\tau})}\right).$$

For any $\epsilon > 0$, there is a parameter setting in which the time it takes to heuristically solve the multivariate Coppersmith problem (excluding the final root recovery step) is polynomial in ϵ^{-1}.

Proof. Equation 3 sets $\boldsymbol{t} = k\boldsymbol{\tau}$ and combines Heuristic 2 with heuristic inequality 1. Since the limit converges polynomially quickly, we may take $k = \Theta(\epsilon^{-1})$ and $\log p = \Theta(\mathrm{poly}(k))$. By Lemma 6, $\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}}$ can be computed in polynomial time. The lattice dimension and entry lengths are polynomial in k, so shift polynomial construction and lattice reduction take polynomial time.

6.6 Precomputing Symbolic Asymptotic Bounds

We combine all ideas in this section into Algorithm 3, which symbolically determines asymptotic bounds for a given multivariate Coppersmith problem. We use Section 6.1 to find a symbolic ideal, Section 6.3 to find $\mathcal{M}_1$, Algorithm 1 to find $\mathcal{S}_1$, Section 6.2 to find symbolic lattice constructions, interpolation to determine s_{dim} and the other polynomials, and Lemma 7 to find asymptotic bounds.

This algorithm, which adapts our methods to the approach of Meers and Nowakowski [33], allows one to compute asymptotic Coppersmith bounds in a fully automated way. While this approach still requires careful choice of $\mathcal{M}_{\mathrm{vert}}$, we believe that it greatly simplifies the process of proving and verifying asymptotic bounds for multivariate Coppersmith problems.

Algorithm 3: Automatically finding multivariate Coppersmith bounds

Input : $\mathcal{F}, \boldsymbol{X}, \boldsymbol{k}_{\text{pre}}, \mathcal{M}_{\text{vert}}, \boldsymbol{X}_{\text{guess}}$
Output: Asymptotic bounds $\boldsymbol{X}$ as determined by δ
Symbolically represent ideals $\hat{J}_p$ and $\hat{J}_\infty$ based on $\mathcal{F}$
$J \leftarrow \hat{J}_p^{\boldsymbol{k}_{\text{pre}}} + \hat{J}_\infty$ for multiplicity $\boldsymbol{k}_{\text{pre}}$
$\mathcal{M}_1 \leftarrow$ integer points in convex hull of $\mathcal{M}_{\text{vert}}$
$\mathcal{S}_1 \leftarrow$ output of Algorithm 1 for J and $\mathcal{M}_1$, triangularized by $<_{\boldsymbol{X}_{\text{guess}}}$
Compute $\mathcal{S}_{k,\boldsymbol{t},\text{ul}}$ for various $(k, \boldsymbol{t})$ using $\mathcal{S}_1$ and $\hat{J}_\infty$
Use polynomial interpolation to recover $s_{\dim}$, $\{s_{x_i}\}_{1 \le i \le \ell}$, and $\{s_{C_j}\}_{C_j \in \text{LC}(S_1)}$
Substitute $\boldsymbol{t} = k\boldsymbol{\tau}$ into bound 3 and consider asymptotic behavior as $k \to \infty$
$\delta \leftarrow$ Maximize the bound over $\boldsymbol{\tau}$
return $(\boldsymbol{\tau}, \delta)$

7 Experiments

We compared the performance of our algorithms to over a dozen different applications of Coppersmith's method, and we report the results in Table 1. It is challenging to directly compare compare two shift polynomial selection strategies, so we therefore focus on the following four attributes to demonstrate the capabilities of our new methods.

- **Bounds.** For a particular multiplicity and shift polynomial strategy, we experimentally determine the maximum $\boldsymbol{X}$ for which Coppersmith's method successfully recovers the expected small root. The provable strategy in Section 4.5 has the strongest guarantees, so we compare the bounds of this strategy to those of prior work.
- **Dimension.** A low-rank lattice is typically faster to reduce. We find that the graph-based strategy in Section 5.2 typically achieves the same bounds as our provable strategy, but involves significantly smaller lattices. We compare the ranks of these lattices to those in prior work.
- **Time.** For practical attacks, running time can be important. We find that our precomputation strategy in Section 6.2 often has similar bounds as our provable strategy and rank as our graph strategy, but can be significantly faster. We compare the concrete running time of this strategy to prior work.
- **Asymptotics.** A regular feature of Coppersmith papers is asymptotic analysis to determine the maximum recoverable bound size as a fraction of the modulus size. We compare the asymptotic analyses from prior work to the automated asymptotic analysis described in Section 6.6.

A number of large-scale trends are apparent in Table 1. First, our provable strategy always performs as well as or better than prior strategies when it comes to maximizing recoverable bounds, supporting our chosen definition of optimality. Second, our graph-based shift polynomial strategy frequently finds smaller, and therefore more easily reduced, lattices. In the cases where this algorithm

Table 1: **Summary of experimental results**. We compare the recoverable bounds, lattice dimension, running time, and asymptotic behavior of our approaches with prior work, and note whether our generalized techniques are better (✔✔), equivalent to (✔), or worse (✗) than prior shift polynomial strategies. We say the strategies are the same if the bounds are within a few bits, the dimension is within a few, or the time is within a factor of two to allow for small variations. Full numerical data supporting these results are in Appendix A.

Coppersmith Problem	Prior	Bounds	Dimension	Time	Asymp.
CIHNP-CSIDH	[33]	✔ same	✔ same	✔ same	✔ same
CIHNP-CSURF	[17]	✔ same	✔✔ better	✔✔ better	✔✔ better
MIHNP	[48]	✔✔ better	✗ worse	✔ same	✔✔ better
ECHNP	[49]	✔✔ better	✗ worse	✗ worse	✔✔ better
Stereotyped RSA	[28]	✔ same	✔ same	✔ same	✔ same
Partial factoring RSA	[28]	✔ same	✔ same	✔ same	✔ same
Partial ACD	[12]	✔ same	✔ same	✔ same	✔ same
Small RSA priv. exp. d	[20]	✔ same	✔✔ better	✔ same	✔ same
RSA Power Gen.	[19]	✔ same	✔✔ better	✔✔ better	✔ same
Small CRT-RSA d_p, d_q	[43]	✔ same	✔✔ better	✗ worse	✗ worse
Partial CRT-RSA d_p, d_q	[30]	✔ same	✔ same	-	-
SMUPE	[32]	✔✔ better	✔✔ better	-	-
Common prime RSA	[25]	✔ same	✔ same	✔ same	✔ same
Small Multipower RSA d	[25]	✔ same	✔ same	✔ same	✔ same

finds a larger lattice, it's because the larger lattice enables recovery with increased bounds. Third, our precomputation strategy is competitive with prior approaches. However, it requires manual identification of monomials $\mathcal{M}_{\text{vert}}$, and poor choice of monomials leads to poor performance. Additionally, our symbolic representation is incompatible with mixed moduli, so we could not apply this strategy in two cases. Fourth, precomputation of asymptotic bounds often leads to the same bounds as in prior work. However, our Algorithm 3 is also sensitive to the choice of $\mathcal{M}_{\text{vert}}$. Sometimes the choice of $\mathcal{M}_{\text{vert}}$ is clear because the monomials found by the graph-based strategy follow a clear pattern as multiplicity increases. In other cases, this is significantly harder. We believe that the problem of choosing $\mathcal{M}_{\text{vert}}$ deserves further study.

Full numerical data and additional details about the diverse array of problems we examined are available in Appendix A. Although there is not room here to examine all aspects, we include experimental results from one of the problems which highlight particular trends in the behavior of our algorithms.

7.1 Experimental setup

We tested the effectiveness of our approaches on several previously studied multivariate Coppersmith problems listed in Table 1. For each problem, we evaluated the maximum bounds $\boldsymbol{X}$ for which the bounded roots are successfully recovered at least 50% of the time for each shift polynomial selection strategy. Each

shift polynomial strategy was evaluated against the same 30 randomly generated problem instances. When a shift polynomial strategy requires additional parameters, we report the parameter that gave the best results. We compare against the provable strategy from Section 4, the graph-based strategy in Section 5, and the precomputation strategy in Section 6. We also report the lattice dimension and average running time in seconds for each multiplicity k.

Each experiment was run in single-threaded mode on a Intel Xeon E5-2699v4 CPU running at 2.2GHz. Our implementation was written in Python. We used SageMath version 10.2 for generic computer algebra tasks, MSolve v0.7.1[6] for Gröbner basis computations in $\mathbb{Q}[\boldsymbol{x}]$, and flatter[7] for lattice reduction.

7.2 Modular Inversion Hidden Number Problem

Boneh, Halevi, and Howgrave-Graham introduced the Modular Inversion Hidden Number Problem (MIHNP) in 2001 [8]. Xu et al. revisited the problem in [48]. For MIHNP, the ℓ input relations have the form

$$f_i(\alpha, x_1, \ldots, x_\ell) = (\alpha + c_{i,1})(x_i + c_{i,2}) - 1$$

and share a bounded root modulo a known prime p.

We report the results for MIHNP with four relations in Table 2. The shift strategy in [48] is only defined for multiplicity $k \leq \ell - 2$, so their existing "asymptotic" bound for four samples is really the maximal bound up to multiplicity 2.

Table 2: **MIHNP with $\ell = 4$ samples**. We use 1000-bit modulus. Hidden number α is 1000-bits, and the unknown values are $\lg X_i$ bits long.

	[48]				All monoms.			Graph search			Precomputed		
k	t	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	0	**332**	7	4.0	**373**	16	0.5	**373**	11	0.4	**-**	-	-
2	1	**405**	16	4.4	**410**	86	12.2	**410**	50	6.9	**405**	16	1.5
3	-	**-**	-	-	**446**	296	211.8	**446**	76	66.0	**-**	-	-
4	-	**-**	-	-	**-**	-	-	**460**	221	581.3	**442**	81	25.7

Existing bound: $\log_p X_i = 0.2432$ **Our bound:** $\log_p X_i = 0.5208$

We ran Algorithm 3 on MIHNP with four samples with input

$$\log_p X_i = \delta, \quad \log_p X_{\text{guess},i} = 0.1, \quad k_{\text{pre}} = 2, \quad \mathcal{M}_{\text{vert}} = \{\prod x_i^{e_i} \mid e_i \in \{0,1\}\}$$

and got output $\tau_i = 0, \delta < 0.5208$.

This example illustrates many interesting features of our approach. First, although alpha is unknown, it is not small. We only have the trivial bound $|\alpha| < p$. While prior works manually manipulated the f_i to eliminate α, our

[6] https://msolve.lip6.fr/
[7] https://github.com/keeganryan/flatter

weighted monomial ordering recognizes $\alpha \gg x_i$, and Algorithm 1 automatically finds shift polynomials with small leading monomial, and therefore exclude α.

This example also illustrates a drawback of our approach. While the approach in [48] provides asymptotic analysis for $\ell \to \infty$, our approach only applies to fixed values of ℓ, and as ℓ increases, Gröbner basis calculations become too expensive.

We also observe that the strategy of using all monomials in $\mathcal{M}_{\text{opt}}$ leads to lattices with large rank and long running times. This is improved by applying the graph-based strategy, which finds smaller lattices with equivalent bounds. However, the graph-based strategy still requires Gröbner basis calculations, and it becomes expensive at high multiplicities as well. The precomputed strategy has worse practical bounds, but it has easily analyzed asymptotic bounds and is fast in practice. While no single strategy is perfect, each has its own benefits when applied to the multivariate Coppersmith problem.

8 Future Work

As a final note, we briefly discuss directions for future work. Although effective for many modular problems, our graph-based methods are ineffective against integer Coppersmith problems where lattices are not full-rank. Our approach also does not capture the multi-step approaches used by Peng et al. [36] or May et al. [30], which construct multiple Coppersmith lattices to gain partial information about the roots. While our work resolves some open questions and greatly simplifies the use of Coppersmith's method, we look forward to seeing future improvements to the capabilities of Coppersmith's method for finding small roots.

Acknowledgments. We thank Nadia Heninger for helpful discussions, and we thank Miro Haller and the anonymous reviewers for providing feedback on drafts of this work. This work was supported in part by a gift from Google.

References

1. Aono, Y.: A new lattice construction for partial key exposure attack for RSA. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 34–53. Springer, Berlin, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00468-1_3
2. Aono, Y., Agrawal, M., Satoh, T., Watanabe, O.: On the optimality of lattices for the Coppersmith technique. In: Susilo, W., Mu, Y., Seberry, J. (eds.) ACISP 12. LNCS, vol. 7372, pp. 376–389. Springer, Berlin, Heidelberg (Jul 2012). https://doi.org/10.1007/978-3-642-31448-3_28
3. Becker, T., Weispfenning, V.: Variations on Gröbner Bases, pp. 453–509. Springer New York, New York, NY (1993). https://doi.org/10.1007/978-1-4612-0913-3_11, https://doi.org/10.1007/978-1-4612-0913-3_11
4. Bleichenbacher, D., May, A.: New attacks on RSA with small secret CRT-exponents. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 1–13. Springer, Berlin, Heidelberg (Apr 2006). https://doi.org/10.1007/11745853_1

5. Blömer, J., May, A.: New partial key exposure attacks on RSA. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 27–43. Springer, Berlin, Heidelberg (Aug 2003). https://doi.org/10.1007/978-3-540-45146-4_2
6. Blömer, J., May, A.: A tool kit for finding small roots of bivariate polynomials over the integers. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 251–267. Springer, Berlin, Heidelberg (May 2005). https://doi.org/10.1007/11426639_15
7. Boneh, D., Durfee, G.: Cryptanalysis of RSA with private key d less than $N^{0.292}$. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 1–11. Springer, Berlin, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_1
8. Boneh, D., Halevi, S., Howgrave-Graham, N.: The modular inversion hidden number problem. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 36–51. Springer, Berlin, Heidelberg (Dec 2001). https://doi.org/10.1007/3-540-45682-1_3
9. Brion, M., Vergne, M.: Lattice points in simple polytopes. Journal of the American Mathematical Society **10**(2), 371–392 (1997), http://www.jstor.org/stable/2152855
10. Chen, B.: Ehrhart polynomials of lattice polyhedral functions (2005). https://doi.org/10.1090/conm/374/06898, http://dx.doi.org/10.1090/conm/374/06898
11. Chinburg, T., Hemenway, B., Heninger, N., Scherr, Z.: Cryptographic applications of capacity theory: On the optimality of Coppersmith's method for univariate polynomials. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016, Part I. LNCS, vol. 10031, pp. 759–788. Springer, Berlin, Heidelberg (Dec 2016). https://doi.org/10.1007/978-3-662-53887-6_28
12. Cohn, H., Heninger, N.: Approximate common divisors via lattices. ANTS X p. 271 (2012). https://doi.org/10.2140/obs.2013.1.271
13. Coppersmith, D.: Finding a small root of a univariate modular equation. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 155–165. Springer, Berlin, Heidelberg (May 1996). https://doi.org/10.1007/3-540-68339-9_14
14. Coppersmith, D.: Finding small solutions to small degree polynomials. In: Silverman, J.H. (ed.) Cryptography and Lattices. pp. 20–31. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
15. Ehrhart, E.: Sur les polyèdres rationnels homothétiques à n dimensions. CR Acad. Sci. Paris **254**, 616 (1962)
16. Ernst, M., Jochemsz, E., May, A., de Weger, B.: Partial key exposure attacks on RSA up to full size exponents. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 371–386. Springer, Berlin, Heidelberg (May 2005). https://doi.org/10.1007/11426639_22
17. Feng, Y., Luo, H., Chen, Q., Nitaj, A., Pan, Y.: Computing asymptotic bounds for small roots in Coppersmith's method via sumset theory. Cryptology ePrint Archive, Paper 2024/1330 (2024), https://eprint.iacr.org/2024/1330
18. Håstad, J.: On using RSA with low exponent in a public key network. In: Williams, H.C. (ed.) CRYPTO'85. LNCS, vol. 218, pp. 403–408. Springer, Berlin, Heidelberg (Aug 1986). https://doi.org/10.1007/3-540-39799-X_29
19. Herrmann, M., May, A.: Attacking power generators using unravelled linearization: When do we output too much? In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 487–504. Springer, Berlin, Heidelberg (Dec 2009). https://doi.org/10.1007/978-3-642-10366-7_29
20. Herrmann, M., May, A.: Maximizing small root bounds by linearization and applications to small secret exponent RSA. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 53–69. Springer, Berlin, Heidelberg (May 2010). https://doi.org/10.1007/978-3-642-13013-7_4

21. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M. (ed.) 6th IMA International Conference on Cryptography and Coding. LNCS, vol. 1355, pp. 131–142. Springer, Berlin, Heidelberg (Dec 1997). https://doi.org/10.1007/bfb0024458
22. Jochemsz, E., May, A.: A strategy for finding roots of multivariate polynomials with new applications in attacking RSA variants. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 267–282. Springer, Berlin, Heidelberg (Dec 2006). https://doi.org/10.1007/11935230_18
23. Jochemsz, E., May, A.: A polynomial time attack on RSA with private CRT-exponents smaller than $N^{0.073}$. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 395–411. Springer, Berlin, Heidelberg (Aug 2007). https://doi.org/10.1007/978-3-540-74143-5_22
24. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen **261**(4), 515–534 (Dec 1982). https://doi.org/10.1007/BF01457454
25. Lu, Y., Zhang, R., Peng, L., Lin, D.: Solving linear equations modulo unknown divisors: Revisited. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 189–213. Springer, Berlin, Heidelberg (Nov / Dec 2015). https://doi.org/10.1007/978-3-662-48797-6_9
26. May, A.: Cryptanalysis of unbalanced RSA with small CRT-exponent. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 242–256. Springer, Berlin, Heidelberg (Aug 2002). https://doi.org/10.1007/3-540-45708-9_16
27. May, A.: New RSA vulnerabilities using lattice reduction methods. Ph.D. thesis, University of Paderborn (2003)
28. May, A.: Using LLL-reduction for solving RSA and factorization problems. pp. 315–348. ISC, Springer (2010). https://doi.org/10.1007/978-3-642-02295-1
29. May, A.: Lattice-based integer factorisation: an introduction to Coppersmith's method. Computational Cryptography: Algorithmic Aspects of Cryptology pp. 78–105 (2021)
30. May, A., Nowakowski, J., Sarkar, S.: Partial key exposure attack on short secret exponent CRT-RSA. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part I. LNCS, vol. 13090, pp. 99–129. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92062-3_4
31. May, A., Nowakowski, J., Sarkar, S.: Approximate divisor multiples - factoring with only a third of the secret CRT-exponents. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 147–167. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_6
32. May, A., Ritzenhofen, M.: Solving systems of modular equations in one variable: How many RSA-encrypted messages does Eve need to know? In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 37–46. Springer, Berlin, Heidelberg (Mar 2008). https://doi.org/10.1007/978-3-540-78440-1_3
33. Meers, J., Nowakowski, J.: Solving the hidden number problem for CSIDH and CSURF via automated Coppersmith. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part IV. LNCS, vol. 14441, pp. 39–71. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8730-6_2
34. Micheli, G.D., Heninger, N.: Survey: Recovering cryptographic keys from partial information, by example. CiC **1**(1), 28 (2024). https://doi.org/10.62056/ahjbksdja
35. Nguyen, P.Q.: Hermite's constant and lattice algorithms. pp. 19–69. ISC, Springer (2010). https://doi.org/10.1007/978-3-642-02295-1

36. Peng, L., Hu, L., Xu, J., Huang, Z., Xie, Y.: Further improvement of factoring RSA moduli with implicit hint. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT 14. LNCS, vol. 8469, pp. 165–177. Springer, Cham (May 2014). https://doi.org/10.1007/978-3-319-06734-6_11
37. Picard, J.C.: Maximal closure of a graph and applications to combinatorial problems. Management Science **22**(11), 1268–1272 (1976). https://doi.org/10.1287/mnsc.22.11.1268, https://doi.org/10.1287/mnsc.22.11.1268
38. Ryan, K., Heninger, N.: Fast practical lattice reduction through iterated compression. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part III. LNCS, vol. 14083, pp. 3–36. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38548-3_1
39. Sarkar, S.: Enhanced bound for the commutative isogeny hidden number problem in CSURF. In: Mukhopadhyay, S., Stănică, P. (eds.) Progress in Cryptology – INDOCRYPT 2024. pp. 201–211. Springer Nature Switzerland, Cham (2025)
40. Sarkar, S., Maitra, S.: Partial key exposure attack on CRT-RSA. In: Abdalla, M., Pointcheval, D., Fouque, P.A., Vergnaud, D. (eds.) ACNS 09International Conference on Applied Cryptography and Network Security. LNCS, vol. 5536, pp. 473–484. Springer, Berlin, Heidelberg (Jun 2009). https://doi.org/10.1007/978-3-642-01957-9_29
41. Shani, B.: On the bit security of elliptic curve Diffie-Hellman. In: Fehr, S. (ed.) PKC 2017, Part I. LNCS, vol. 10174, pp. 361–387. Springer, Berlin, Heidelberg (Mar 2017). https://doi.org/10.1007/978-3-662-54365-8_15
42. Takayasu, A., Kunihiro, N.: Partial key exposure attacks on RSA: Achieving the boneh-durfee bound. In: Joux, A., Youssef, A.M. (eds.) SAC 2014. LNCS, vol. 8781, pp. 345–362. Springer, Cham (Aug 2014). https://doi.org/10.1007/978-3-319-13051-4_21
43. Takayasu, A., Lu, Y., Peng, L.: Small CRT-exponent RSA revisited. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part II. LNCS, vol. 10211, pp. 130–159. Springer, Cham (Apr / May 2017). https://doi.org/10.1007/978-3-319-56614-6_5
44. Xu, J., Hu, L., Huang, Z., Peng, L.: Modular inversion hidden number problem revisited. In: Huang, X., Zhou, J. (eds.) Information Security Practice and Experience. pp. 537–551. Springer International Publishing, Cham (2014)
45. Xu, J., Hu, L., Sarkar, S.: Cryptanalysis of elliptic curve hidden number problem from PKC 2017. DCC **88**(2), 341–361 (2020). https://doi.org/10.1007/s10623-019-00685-y
46. Xu, J., Sarkar, S., Hu, L., Huang, Z., Peng, L.: Solving a class of modular polynomial equations and its relation to modular inversion hidden number problem and inversive congruential generator. DCC **86**(9), 1997–2033 (2018). https://doi.org/10.1007/s10623-017-0435-4
47. Xu, J., Sarkar, S., Hu, L., Wang, H., Pan, Y.: New results on modular inversion hidden number problem and inversive congruential generator. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 297–321. Springer, Cham (Aug 2019). https://doi.org/10.1007/978-3-030-26948-7_11
48. Xu, J., Sarkar, S., Hu, L., Wang, H., Pan, Y.: Revisiting modular inversion hidden number problem and its applications. IEEE Transactions on Information Theory **69**(8), 5337–5356 (2023). https://doi.org/10.1109/TIT.2023.3263485
49. Xu, J., Sarkar, S., Wang, H., Hu, L.: Improving bounds on elliptic curve hidden number problem for ECDH key exchange. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part III. LNCS, vol. 13793, pp. 771–799. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22969-5_26

A Experimental Data

A.1 Commutative Isogeny Hidden Number Problem

The Commutative Isogeny Hidden Number Problem (CI-HNP) was proposed by Meers and Nowakowski to study the bit security of isogeny-based key exchange schemes [33]. The problem asks whether it is possible to recover a shared elliptic curve based on the most significant bits of a Diffie-Hellman style key. They examine CI-HNP for both CSIDH and CSURF key exchanges.

For CSIDH, the input relations have the form

$$\begin{aligned} f &= (c_1 + x_1)(c_2 + x_2) + 2(c_1 + x_1) - 2(c_2 + x_2) + 12 \\ g &= (c_3 + x_3)(c_2 + x_2) + 2(c_2 + x_2) - 2(c_3 + x_3) + 12 \\ h &= (c_1 + x_1)(c_3 + x_3) - 2(c_1 + x_1) + 2(c_3 + x_3) + 12 \end{aligned}$$

and share a root modulo a prime p. For CSURF, the input relations are

$$\begin{aligned} f &= (c_1 + x_1)^2 + 12(c_1 + x_1) - 4(c_1 + x_1)(c_2 + x_2)^2 - 8(c_2 + x_2)^2 + 36 \\ g &= (c_3 + x_3)^2 + 12(c_3 + x_3) - 4(c_3 + x_3)(c_1 + x_1)^2 - 8(c_1 + x_1)^2 + 36. \end{aligned}$$

There has been recent progress improving the bounds for CIHNP-CSURF. The original method of Meers and Nowakowski solved up to $\log_p X_i = 0.2439$, and this was improved by an early draft of Feng et al. [17] in October 2024 to $\log_p X_i = 0.2580$. A draft of this work from October 2024 identified $\mathcal{M}_{\text{vert}}$ achieving $\log_p X_i = 0.2528$, a slightly worse bound. Sarkar improved this to $\log_p X_i = 0.2597$ in a December 2024 publication [39]. A February 2025 draft by Feng et al. [17] improved this to $\log_p X_i = 0.2599$. The current version of this work achieves $\log_p X_i = 0.2609$.

We ran our shift polynomial strategies on CI-HNP for both CSIDH and CSURF, and report the results in Table 3 and Table 4 respectively.

Table 3: **CI-HNP for CSIDH**. Largest solvable bound for 512-bit modulus p. Note that [33] is only compatible if the multiplicity is divisible by 3.

	[33]			All monoms.			Graph search			Precomputed		
k	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	**-**	-	-	**169**	20	0.5	**169**	8	0.5	**-**	-	-
2	**-**	-	-	**189**	56	3.4	**189**	26	2.4	**-**	-	-
3	**194**	27	1.7	**199**	120	25.3	**199**	60	16.3	**194**	27	2.3
4	**-**	-	-	**204**	286	248.9	**204**	115	87.0	**-**	-	-
5	**-**	-	-	**210**	455	1029.1	**210**	125	169.2	**-**	-	-
6	**210**	125	98.9	**-**	-	-	**213**	215	528.2	**210**	125	94.4
7	**-**	-	-	**-**	-	-	**215**	339	1406.6	**-**	-	-
8	**-**	-	-	**-**	-	-	**216**	502	3158.2	**-**	-	-
9	**216**	343	1871.6	**-**	-	-	**-**	-	-	**216**	343	1178.9

Existing bound: $\log_p X_i = 0.4583$ **Our bound:** $\log_p X_i = 0.4583$

We ran Algorithm 3 on CIHNP-CSIDH with input

$$\log_p X_i = \delta, \quad \log_p X_{\text{guess},i} = 0.1, \quad k_{\text{pre}} = 3, \quad \mathcal{M}_{\text{vert}} = \{\prod_{i=1}^{3} x_i^{e_i} \mid e_i \in \{0, 2\}\}$$

and got output

$$\boldsymbol{\tau} = (0, 0, 0), \quad \delta < 0.4583.$$

This matches the existing bounds of $X_i < p^{11/24}$ [33].

Table 4: **CI-HNP for CSURF**. Largest solvable bound for 512-bit modulus p. In the all-monomial strategy with $k = 2$, we have $X^{11} \approx p^2$, so whichever value is larger determines $|\mathcal{M}_{\text{opt}}|$. We report the maximum size, which is 364. $\boldsymbol{t}$ refers to the parameters (t_1, t_2) as defined in [17].

	[17]				All monoms.			Graph search			Precomputed		
k	$\boldsymbol{t}$	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	1,0	**71**	103	11.2	**71**	120	14.1	**69**	48	8.5	**-**	-	-
2	1,0	**93**	229	198.0	**93**	364	291.9	**91**	145	142.7	**82**	65	14.1
3	1,0	**103**	427	1588.5	**103**	680	3513.2	**101**	324	1333.2	**-**	-	-
4	1,0	**109**	713	9610.9	**-**	-	-	**-**	-	-	**105**	369	1127.5
5	1,0	**113**	1103	38759.2	**-**	-	-	**-**	-	-	**-**	-	-
6	-	**-**	-	-	**-**	-	-	**-**	-	-	**114**	1105	32036.3

Existing bound: $\log_p X_i = 0.2599$ **Our bound:** $\log_p X_i = 0.2609$

We ran Algorithm 3 on CIHNP-CSURF with input

$$\log_p X_i = \delta, \quad \log_p X_{\text{guess},i} = 0.1, \quad k_{\text{pre}} = 2,$$

$$\mathcal{M}_{\text{vert}} = \{1, x_1^4, x_1^4 x_3^2, x_3^4, x_2^4, x_1^2 x_2^4, x_1^2 x_2^4 x_3^2, x_2^4 x_3^2\}$$

and got output

$$\boldsymbol{\tau} = (0, 0, 0), \quad \delta < 0.2609.$$

This bound of $\delta < 451/1728$ improves Feng et al.'s bound [17] of $X_i < p^{0.2599}$.

A.2 Modular Inversion Hidden Number Problem

Boneh, Halevi, and Howgrave-Graham introduced the Modular Inversion Hidden Number Problem (MIHNP) in 2001 [8]. Xu et al. revisited the problem in [48]. For MIHNP, the ℓ input relations have the form

$$f_i(\alpha, x_1, \ldots, x_\ell) = (\alpha + c_{i,1})(x_i + c_{i,2}) - 1$$

and share a root modulo a known prime p.

Because the bound A on α is large ($A = p$), prior approaches use resultants to manually construct input relations that do not include α. However, since our

monomial ordering includes the size of α, our Gröbner basis strategies automatically find shift polynomials that do not include α.

We ran our shift polynomial strategies on MIHNP for three, four, and five relations, and we report the results in Table 5, Table 6, and Table 7 respectively. The shift strategy in [48] is parameterized by (n', d', t'), which using our variable names is $(\ell - 1, k, t)$. Their strategy is only defined for $0 \leq t \leq k$ and $k \leq \ell - 2$, so their result is not asymptotic in multiplicity k. To compute the existing theoretical bound, we evaluate $\max_{k,t} \hat{\lambda}(\ell - 1, k, t)$ where $\hat{\lambda}$ is defined in [48, Section VI.A].

Table 5: **MIHNP with $\ell = 3$ samples**.

	[48]				All monoms.			Graph search			Precomputed		
k	t	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	0	**248**	4	4.6	**331**	21	0.6	**332**	8	0.4	**-**	-	-
2	-	**-**	-	-	**371**	67	5.6	**372**	26	3.7	**372**	27	4.4
3	-	**-**	-	-	**390**	187	85.6	**390**	60	27.8	**-**	-	-
4	-	**-**	-	-	**401**	386	837.9	**401**	115	152.1	**401**	125	74.4

Existing bound: $\log_p X_i = 0.0000$ **Our bound:** $\log_p X_i = 0.4444$

Table 6: **MIHNP with $\ell = 4$ samples**.

	[48]				All monoms.			Graph search			Precomputed		
k	t	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	0	**332**	7	4.0	**373**	16	0.5	**373**	11	0.4	**-**	-	-
2	1	**405**	16	4.4	**410**	86	12.2	**410**	50	6.9	**405**	16	1.5
3	-	**-**	-	-	**446**	296	211.8	**446**	76	66.0	**-**	-	-
4	-	**-**	-	-	**-**	-	-	**460**	221	581.3	**442**	81	25.7

Existing bound: $\log_p X_i = 0.2432$ **Our bound:** $\log_p X_i = 0.5208$

Table 7: **MIHNP with $\ell = 5$ samples**.

	[48]				All monoms.			Graph search			Precomputed		
k	t	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	0	**373**	11	5.0	**398**	22	1.0	**398**	16	1.0	**-**	-	-
2	1	**449**	30	5.6	**451**	148	40.5	**451**	31	10.1	**451**	32	9.9
3	0	**451**	46	5.4	**480**	610	3136.8	**480**	192	312.3	**-**	-	-
4	-	**-**	-	-	**-**	-	-	**496**	303	2408.5	**487**	243	474.2

Existing bound: $\log_p X_i = 0.3708$ **Our bound:** $\log_p X_i = 0.5577$

We ran Algorithm 3 on MIHNP with $\ell \in \{3, 4, 5\}$ samples with input

$$\log_p A = 1, \quad \log_p X_i = \delta, \quad \log_p X_{\text{guess},i} = 0.1, \quad k_{\text{pre}} = 2,$$

$$\mathcal{M}_{\text{vert}} = \begin{cases} \{\prod_{i=1}^{\ell} x_i^{e_i} \mid e_i \in \{0, 2\}\} & \ell = 3 \\ \{\prod_{i=1}^{\ell} x_i^{e_i} \mid e_i \in \{0, 1\}\} & \ell \in \{4, 5\} \end{cases}$$

and got output

$$\tau_i = 0, \quad \delta < \begin{cases} 0.4444 & \ell = 3 \\ 0.5208 & \ell = 4 \\ 0.5577 & \ell = 5. \end{cases}$$

A.3 Elliptic Curve Hidden Number Problem

The Elliptic Curve Hidden Number Problem (ECHNP) was proposed by Shani in 2017 [41] and studied by Xu et al. in 2022 [49]. The problem studies the hardness of recovering a shared Elliptic Curve Diffie Hellman secret from an oracle that computes most significant bits.

For ECHNP, the $\ell - 1$ input relations have the form

$$x_1^2 x_{i+1} + E_i x_1 x_{i+1} + D_i x_{i+1} + C_i x_1^2 + B_i x_1 + A_i \equiv 0 \pmod{p}$$

where known values $\{a, b, h_0, h_1, \ldots, h_{\ell-1}, x_{Q_1}, \ldots x_{Q_{\ell-1}}\}$ are used to compute

$$\begin{aligned} A_i &= h_i(h_0 - x_{Q_i})^2 - 2h_0^2 x_{Q_i} - 2(a + x_{Q_i}^2)h_0 - 2a x_{Q_i} - 4b \\ B_i &= 2(h_i(h_0 - x_{Q_i}) - 2h_0 x_{Q_i} - a - x_{Q_i}^2) \\ C_i &= h_i - 2x_{Q_i} \\ D_i &= (h_0 - x_{Q_i})^2 \\ E_i &= 2(h_0 - x_{Q_i}), \end{aligned}$$

and the relations share a root modulo a known prime p.

We ran our shift polynomial strategies on ECHNP for three, four, and five relations, and we report the results in Tables 8, 9, and 10 respectively. The shift strategy in [49] is parameterized by (n', d', t'), which using our variable names is (ℓ, k, t). Their strategy is only defined for $1 \leq k < \ell$ and $0 \leq t \leq 2k - 1$, so their result is not asymptotic in multiplicity k. To compute the existing theoretical bound, we evaluate $\max_{k,t} S(\ell, k, t)$ where S is defined in [49, Section 4.2].

Table 8: **ECHNP with 3 samples**. We consider 256-bit modulus p. The shift strategy in [49] is parameterized by (n', d', t'), which using our variable names is $(3, k, t)$.

	[49]				All monoms.			Graph search			Precomputed		
k	t	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	1	**60**	14	0.6	**64**	35	1.8	**64**	19	1.3	**58**	19	5.9
2	3	**68**	32	2.4	**78**	210	102.2	**78**	111	58.1	**64**	108	33.0
3	2	**68**	48	3.3	**85**	715	2418.1	**83**	397	1217.1	**66**	366	569.8

Existing bound: $\log_p X_i = 0.2083$ **Our bound:** $\log_p X_i = 0.3090$

Table 9: **ECHNP with 4 samples**. We consider 256-bit modulus p. The shift strategy in [49] is parameterized by (n', d', t'), which using our variable names is $(4, k, t)$. We do not run our precomputation strategy because computing the symbolic Gröbner basis was too expensive with the large number of variables.

		[49]			All monoms.			Graph search		
k	2	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	1	**69**	22	1.3	**72**	56	3.5	**72**	26	2.9
2	3	**79**	60	9.8	**87**	252	215.1	**87**	172	130.3

Existing bound: $\log_p X_i = 0.2772$

Table 10: **ECHNP with 5 samples**. We consider 256-bit modulus p. The shift strategy in [49] is parameterized by (n', d', t'), which using our variable names is $(5, k, t)$. We do not run our precomputation strategy because computing the symbolic Gröbner basis was too expensive with the large number of variables.

		[49]			All monoms.			Graph search		
k	2	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	1	**75**	32	2.3	**77**	84	8.7	**78**	42	5.9
2	2	**87**	94	21.9	**92**	462	1301.5	**92**	309	895.2

Existing bound: $\log_p X_i = 0.3224$

We ran Algorithm 3 on ECHNP with 3 samples with input

$$\log_p X_i = \delta, \quad \log_p X_{\text{guess},i} = 0.1, \quad k_{\text{pre}} = 1,$$

$$\mathcal{M}_{\text{vert}} = \{1, x_1^2, x_2^2, x_3^2, x_4^2, x_1^2x_2^2, x_1^2x_3^2, x_1^2x_4^2, x_2x_3x_4\}$$

and got output

$$\tau_i = 0, \quad \delta < 0.3090.$$

This improves on the previous asymptotic bounds for three ECHNP samples.

A.4 Stereotyped RSA

One of Coppersmith's original applications was recovering an RSA plaintext from fixed affine padding [13]. Given modulus N, padding a and ciphertext c, recover bounded message x by solving the modular polynomial The input relation is therefore

$$f(x) = (a + x)^3 - c \equiv 0 \pmod{N}.$$

We use an alternative shift polynomial strategy by May [28] to solve this problem. Although this is a univariate example, it serves as a baseline to validate our approach. We use the shift polynomial strategy in [28], which is parametrized by the multiplicity k and parameter t. We ran our shift polynomial strategies on the stereotyped RSA problem and report the results in Tables 11.

Table 11: **Stereotyped RSA** Largest solvable bounds for 1000-bit N.

		[28]			All monoms.			Graph search			Precomputed		
k	t	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time
1	2	**199**	5	0.0	**199**	6	0.1	**199**	6	0.2	**166**	4	0.1
2	2	**249**	8	0.1	**249**	9	0.3	**249**	9	0.2	**237**	7	0.1
4	2	**285**	14	0.4	**285**	15	0.7	**285**	15	0.8	**281**	13	0.4
6	2	**299**	20	0.9	**299**	21	1.6	**299**	21	1.6	**297**	19	0.9
8	2	**307**	26	1.7	**307**	27	3.2	**307**	27	3.2	**306**	25	1.8
10	2	**312**	32	2.6	**312**	33	5.6	**312**	33	5.7	**311**	31	3.0
12	2	**315**	38	4.6	**315**	39	9.7	**315**	39	9.7	**314**	37	5.0
14	2	**317**	44	7.2	**317**	45	16.1	**317**	45	16.1	**317**	43	7.5
16	2	**319**	50	10.7	**319**	51	23.5	**319**	51	24.4	**319**	49	11.1
18	2	**321**	56	12.9	**321**	57	33.9	**321**	57	34.6	**320**	55	14.3
20	2	**322**	62	17.0	**322**	63	52.7	**322**	63	55.8	**322**	61	19.4

Existing bound: $\log_N X = 0.3333$ **Our bound:** $\log_N X = 0.3333$

We ran Algorithm 3 on the stereotyped RSA problem with input

$$\log_N X = \delta, \quad \log_N X_{\text{guess},i} = 0.1, \quad k_{\text{pre}} = 1, \quad \mathcal{M}_{\text{vert}} = \{1, x^3\}$$

and got output

$$\tau = 0, \quad \delta < 0.3333.$$

This agrees with existing asymptotic bounds of $X < N^{1/3}$ [13].

Observe that our strategies of using all monomials or graph search are competitive with the previous strategy, but the cost of Gröbner basis calculation is problematic at high multiplicity. The precomputed strategy avoids this cost, and even though it performs worse at low multiplicities, it matches the existing asymptotic behavior.

A.5 RSA factoring with high bits known

We examine the problem of factoring RSA modulus $N = pq$ when the most significant bits of p are known. This involves modular relations

$$\begin{aligned} N &\equiv 0 \pmod{p} \\ x + a &\equiv 0 \pmod{p}. \end{aligned}$$

We use the shift polynomial strategy by May [28], which is parametrized by the multiplicity k and parameter t. We ran our shift polynomial strategies on this problem and report the results in Table 12.

Table 12: **RSA partial factoring** Largest solvable bounds for 2048-bit N

	[28]				All monoms.			Graph search			Precomputed		
k	t	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time
1	2	**340**	3	0.0	**340**	4	0.1	**340**	4	0.0	**340**	3	0.1
5	6	**464**	11	0.3	**464**	12	0.5	**464**	12	0.5	**464**	11	0.4
10	11	**486**	21	1.9	**486**	22	3.1	**486**	22	2.9	**486**	21	2.1
15	16	**494**	31	7.7	**494**	32	12.3	**494**	32	11.5	**494**	31	9.1
20	21	**498**	41	18.4	**498**	42	29.0	**498**	42	28.5	**498**	41	22.9
25	25	**500**	50	56.4	**500**	52	63.8	**500**	52	72.9	**500**	51	60.4

Existing bound: $\log_p X = 0.5$ **Our bound:** $\log_p X = 0.5$

We ran Algorithm 3 on the RSA partial factoring problem with

$$\log_p X = \delta, \quad \log_p X_{\text{guess}} = 0.1, \quad k_{\text{pre}} = 1, \quad \mathcal{M}_{\text{vert}} = \{1, x\}$$

and got output

$$\tau = 1.0000, \quad \delta < 0.5000.$$

This agrees with existing asymptotic bounds of $X < N^{1/4}$ [28].

A.6 Partial Approximate Common Divisors

Heninger and Cohn studied the Partial Approximate Common Divisors (PACD) problem in [12]. Input relations are defined modulo p, and a multiple N of p is known. PACD also involves ℓ samples c_i which are close to multiples of p. The input relations are therefore

$$\begin{cases} N \equiv 0 \pmod{p} \\ c_i - x_i \equiv 0 \pmod{p} \quad 1 \le i \le \ell. \end{cases}$$

We ran our shift polynomial strategies on PACD for $\ell \in \{1, 2, 3\}$ samples and report the results in Tables 13, 14, and 15 respectively. In all cases, we used 1000-bit N and 400-bit p.

Table 13: **PACD with $\ell = 1$ sample**.

	[12]				All monoms.			Graph search			Precomputed		
k	t	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	3	**99**	4	0.0	**99**	5	0.3	**99**	5	0.1	**99**	4	0.3
5	12	**140**	13	0.4	**142**	15	1.0	**142**	15	0.8	**142**	14	0.5
10	25	**150**	26	3.6	**150**	27	4.9	**150**	27	4.8	**150**	26	4.0
15	37	**153**	38	11.9	**153**	40	15.3	**153**	40	15.2	**153**	39	12.6
20	50	**155**	51	27.6	**155**	52	42.3	**155**	52	42.3	**155**	51	35.7
25	61	**155**	62	90.3	**156**	65	112.9	**156**	65	112.2	**156**	64	88.2

Existing bound: $\log_p X_i = 0.4000$ **Our bound:** $\log_p X_i = 0.4000$

Table 14: **PACD with $\ell = 2$ samples**.

		[12]			All monoms.			Graph search			Precomputed		
k	t	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	2	**174**	6	0.1	**174**	6	0.3	**174**	6	0.2	**-**	-	-
3	5	**216**	21	1.1	**216**	21	1.4	**216**	21	1.4	**216**	26	1.5
6	10	**231**	66	20.8	**231**	66	23.1	**231**	66	23.0	**231**	64	20.2
9	14	**237**	120	164.6	**237**	136	213.1	**237**	136	215.5	**237**	147	208.6
12	19	**241**	210	753.7	**241**	210	949.4	**241**	210	946.4	**241**	225	925.2

Existing bound: $\log_p X_i = 0.6324$ **Our bound:** $\log_p X_i = 0.6321$

Table 15: **PACD with $\ell = 3$ samples**.

		[12]			All monoms.			Graph search			Precomputed		
k	t	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	1	**198**	4	0.1	**198**	10	0.5	**198**	10	0.2	**-**	-	-
2	3	**243**	20	0.9	**243**	20	1.3	**243**	20	1.1	**-**	-	-
3	4	**256**	35	4.0	**256**	35	4.5	**256**	35	4.6	**255**	96	7.4
4	6	**261**	84	26.9	**261**	84	32.7	**261**	84	28.1	**-**	-	-
5	7	**268**	120	67.4	**268**	120	104.9	**268**	120	91.2	**-**	-	-
6	8	**271**	165	293.7	**271**	165	298.9	**271**	165	298.3	**271**	328	451.1

Existing bound: $\log_p X_i = 0.7368$ **Our bound:** $\log_p X_i = 0.7368$

We ran Algorithm 3 on PACD with $\ell = 1$ sample with input

$$\log_p X_i = \delta, \quad \log_p X_{\text{guess},i} = 0.1, \quad k_{\text{pre}} = 1, \quad \mathcal{M}_{\text{vert}} = \{1, x_1\}$$

and got output

$$\tau_i = 1.5000, \quad \delta < 0.4000.$$

We also ran Algorithm 3 on PACD with $\ell \in \{2, 3\}$ samples with input

$$\log_p X_i = \delta, \quad \log_p X_{\text{guess},i} = 0.1, \quad k_{\text{pre}} = 3, \quad \mathcal{M}_{\text{vert}} = \{1, x_1^4, x_2^4, \ldots, x_\ell^4\}$$

and got output

$$\begin{cases} \tau_i = 0.3811, & \delta < 0.6321 \quad \ell = 2 \\ \tau_i = 0.0240, & \delta < 0.7368 \quad \ell = 3. \end{cases}$$

This is nearly the previous bound of $\log_p \delta < (\log_N p)^{1/\ell}$.

A.7 RSA with small private exponent

Boneh and Durfee showed that recovering a small RSA private exponent from a public key (N, e) is possible by solving the small inverse problem [7]. In particular, they consider the relation

$$x_1 x_2 - x_1(N+1) - 1 \equiv 0 \pmod{e}$$

which has small modular root (r_1, r_2) with $|r_1| < e^{\delta}$ and $|r_2| < e^{1/2}$. In 2010, Herrmann and May used unravelled linearization

$$u = x_1 x_2 - 1$$

to simplify analysis of the problem's solveable bounds [20].

We ran our shift polynomial strategies on the RSA with small private exponent problem and report the results in Table 16.

Table 16: **RSA with small private exponent**. Maximally recoverable bound for size of RSA private exponent for 1000-bit modulus and full-size exponent. In our strategies with $k = 7$, variations in modulus e lead to variations in $|\mathcal{M}_{\text{opt}}|$. We report the maximum size for our all-monomial and graph strategy, which are 746 and 42 respectively.

		[20]			All monoms.			Graph search			Precomputed		
k	t	$\lg X_1$	Dim.	Time	$\lg X_1$	Dim.	Time	$\lg X_1$	Dim.	Time	$\lg X_1$	Dim.	Time
2	1	**243**	7	0.3	**240**	40	1.5	**-**	-	-	**-**	-	-
3	1	**259**	11	0.3	**259**	83	7.7	**259**	8	0.9	**259**	15	0.5
4	2	**263**	19	1.1	**263**	174	66.1	**263**	14	2.3	**-**	-	-
5	2	**267**	27	1.8	**267**	308	432.0	**267**	22	3.8	**-**	-	-
6	3	**270**	37	3.8	**270**	493	2391.7	**270**	31	7.2	**270**	40	3.5
7	3	**272**	48	7.5	**271**	746	5418.1	**272**	42	13.3	**-**	-	-
8	3	**274**	60	12.1	**-**	-	-	**274**	50	22.0	**-**	-	-
9	4	**275**	75	24.7	**-**	-	-	**275**	63	39.9	**275**	77	22.1

Existing bound: $\log_N X_1 = 0.2928$ **Our bound:** $\log_N X_1 = 0.2925$

We ran Algorithm 3 on the small RSA private exponent problem with input

$$\log_N(X_1, X_2, U) = (\delta, \frac{1}{2}, \frac{1}{2} + \delta), \quad \log_N \boldsymbol{X}_{\text{guess}} = (0.1, 0.5, 0.6), \quad k_{\text{pre}} = 3,$$

$$\mathcal{M}_{\text{vert}} = \{1, x_1^3, u^3, x_2 u^3\}$$

and got output

$$\boldsymbol{\tau} = (0, 0.1230, 0), \quad \delta < 0.2925.$$

This is nearly the existing bound of $\log_N X_i < 1 - \frac{\sqrt{2}}{2}$.

Note that $\tau_2 > 0$; our algorithm automatically found the x_2-shifts that were central to [7]. Our automatically determined bound is slightly worse, but we note that a higher precomputed multiplicity gets even closer, and speculate that the gap may be related to the irrationality of $1 - \frac{\sqrt{2}}{2}$.

Our performance is comparable to [20], but observe that graph search finds a smaller sublattice.

A.8 RSA Power Generators

Herrmann and May studied the problem of state recovery attacks on RSA-based random number generators [19]. In this problem, an adversary is given the most-significant bits c_i of states obtained from repeated exponentiation. The task is to recover the unknown least-significant bits. For public modulus N and ℓ outputs, this yields the relations

$$(x_i + c_i)^2 - (x_{i+1} + c_{i+1}) \equiv 0 \pmod{N} \quad \text{for } 1 \leq i \leq \ell - 1.$$

Herrmann and May introduce the concept of unravelled linearization, adding

$$x_i^2 - u_i - x_{i+1} = 0 \quad \text{for } 1 \leq i \leq \ell - 1.$$

We ran our shift polynomial strategies on the Power Generator state recovery problem for $\ell \in \{2, 3\}$, and report the results in Table 17 and Table 18 respectively.

Table 17: **RSA Power Generators with $\ell = 2$ samples** Largest bit leakage leading to recovery of RSA RNG states with 1024-bit modulus.

	[19]			All monoms.			Graph search			Precomputed		
k	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	**340**	3	0.1	**339**	13	0.5	**340**	3	0.2	**-**	-	-
2	**371**	6	0.2	**371**	34	1.5	**371**	6	0.4	**371**	6	0.3
3	**378**	16	1.0	**378**	95	16.5	**378**	10	1.0	**-**	-	-
4	**385**	25	0.8	**385**	161	97.1	**385**	15	2.0	**385**	15	0.8
5	**388**	36	7.2	**388**	308	785.6	**388**	21	4.2	**-**	-	-
6	**392**	49	22.5	**391**	444	2744.1	**391**	28	7.3	**391**	28	2.6

Existing bound: $\log_N X_i = 0.4000$ **Our bound:** $\log_N X_i = 0.4000$

We ran Algorithm 3 on the RSA Power Generator problem with input

$$\log_N X_i = \delta, \quad \log_N U_i = 2\delta, \quad \log_N X_{\text{guess},i} = 0.1, \quad k_{\text{pre}} = 2,$$

$$\mathcal{M}_{\text{vert}} = \{1, x_1^2, x_2, u_1^2\}$$

and got output

$$\boldsymbol{\tau} = (0, 0, 0), \quad \delta < 0.4000.$$

This matches the existing bounds of $X_i < N^{2/5}$ for two outputs [19].

Table 18: **RSA Power Generators with $\ell = 3$ samples** Largest bit leakage leading to recovery of RSA RNG states with 1024-bit modulus.

	[19]			All monoms.			Graph search			Precomputed		
k	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time	$\lg X_i$	Dim.	Time
1	**-**	-	-	**339**	28	1.0	**340**	6	0.4	**-**	-	-
2	**392**	7	0.3	**392**	108	15.5	**392**	7	1.3	**-**	-	-
3	**397**	22	1.7	**402**	308	459.7	**402**	21	5.0	**-**	-	-
4	**413**	39	1.7	**-**	-	-	**413**	22	16.3	**413**	22	2.2
5	**419**	62	10.5	**-**	-	-	**423**	50	77.9	**-**	-	-
6	**427**	93	61.2	**-**	-	-	**429**	93	172.2	**-**	-	-
7	**430**	132	448.5	**-**	-	-	**433**	95	369.7	**-**	-	-
8	**433**	181	5910.4	**-**	-	-	**437**	159	862.7	**435**	95	50.7

Existing bound: $\log_N X_i = 0.4615$ **Our bound:** $\log_N X_i = 0.4615$

We ran Algorithm 3 on the RSA Power Generator problem with input

$$\log_N X_i = \delta, \quad \log_N U_i = 2\delta, \quad \log_N X_{\text{guess},i} = 0.1, \quad k_{\text{pre}} = 4,$$

$$\mathcal{M}_{\text{vert}} = \{1, x_1^4, x_2^2, x_3, u_1^4, u_2^2\}$$

and got output

$$\boldsymbol{\tau} = (0, 0, 0, 0, 0), \quad \delta < 0.4615.$$

This matches the existing bounds of $X_i < N^{6/13}$ for three outputs [19].

A.9 RSA-CRT with small private exponents

We consider the problem of RSA-CRT with small private exponents, first explored in [26], with the best current results due to Takayasu, Lu, and Peng [43]. The problem considers RSA modulus $N = pq$ with public exponent e and small private exponents (d_p, d_q) satisfying $ed_p \equiv 1 \pmod{p-1}$ and $ed_q \equiv 1 \pmod{q-1}$. This is rewritten as the following relations

$$\begin{aligned} -1 - x_3(x_1 - 1) &\equiv 0 \pmod{e} \\ -1 - x_4(x_2 - 1) &\equiv 0 \pmod{e} \\ x_1 x_2 - N &= 0. \end{aligned}$$

with shared root $(p, q, \frac{ed_p-1}{p-1}, \frac{ed_q-1}{q-1})$ and bounds $\boldsymbol{X} = (1/2, 1/2, 1/2+\delta, 1/2+\delta)$ for $e \approx N$. Thus $d_p \approx X_3/X_1$. We introduce the unravelled linearization

$$\begin{aligned} u_1 &= x_1 + x_2 & u_2 &= x_3x_1 + x_4x_2 + 2 \\ u_3 &= x_3 + x_4 - 1 & u_4 &= x_3x_4 \\ u_5 &= x_3x_4x_1 + x_3x_4x_2 - x_3x_1 - x_4x_2 + x_3 + x_4 - 1 \end{aligned}$$

and ran our shift polynomial strategies on the problem of RSA with small CRT exponents and report the results in Table 19.

Table 19: **RSA-CRT with small secret exponents**. Maximum bound for private exponents d_p and d_q for 1000-bit modulus N and full-size prime e. The strategy of [43] yields a lattice of dimension 177 or 179 for $k = 8$, so we report the smaller value.

	[43]			Graph search			Precomputed		
k	$\lg \frac{X_3}{X_1}$	Dim.	Time	$\lg \frac{X_3}{X_1}$	Dim.	Time	$\lg \frac{X_3}{X_1}$	Dim.	Time
4	**33**	31	5.9	**33**	15	42.2	**33**	21	1.8
5	**33**	31	3.7	**39**	40	174.3	**-**	-	-
6	**52**	84	28.8	**51**	42	779.8	**-**	-	-
7	**52**	84	30.0	**56**	88	2204.6	**-**	-	-
8	**62**	177	308.5	**62**	89	7189.0	**43**	102	33.3

Existing bound: $\log_N \frac{X_3}{X_1} = 0.1220$ **Our bound:** $\log_N \frac{X_3}{X_1} = 0.0468$

We ran Algorithm 3 on the small RSA-CRT exponent problem with input

$$\log_N \boldsymbol{X} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}+\delta, \frac{1}{2}+\delta), \quad \log_N \boldsymbol{U} = (\frac{1}{2}, 1+\delta, \frac{1}{2}+\delta, 1+2\delta, \frac{3}{2}+2\delta),$$

$$\log_N \boldsymbol{X}_{\text{guess}} = (0.5, 0.5, 0.65, 0.65), \quad k_{\text{pre}} = 4,$$

$$\mathcal{M}_{\text{vert}} = \{1, u_1, u_2^2, u_3^2, u_4^2, u_5^2, u_1u_3^2, u_1u_5^2, u_2u_3, u_2u_4, u_2u_5\}$$

We fixed $\boldsymbol{\tau} = 0$ because Algorithm 3 was too slow otherwise and got output

$$\delta < 0.0486.$$

This is worse than the existing bound of $\log_N \frac{X_3}{X_1} < \frac{1}{2} - \frac{1}{\sqrt{7}}$.

As seen in Table 19, even though $\mathcal{M}_{\text{opt}}$ was too large to run our strategy on all monomials, our graph-based search algorithm found sublattices of approximately half the dimension of [43]. Compared to the complicated multi-page shift polynomial selection strategy in prior works, this demonstrates the value and effectiveness of our automated approach.

A.10 Partial Key Exposure attacks on CRT-RSA

In 2021, May, Nowakowski, and Sarkar studied Partial Key Exposure attacks on CRT-RSA [30]. They analyze the case of RSA public exponent e that scales with modulus $N = pq$. An attacker learns the least-significant (or most-significant) bits of private CRT exponents d_p, d_q. May et al.'s strategy has two steps: first, recover $(ed_p - 1)/(p-1)$ and $(ed_q - 1)/(q-1)$ using a Coppersmith-style attack (for the case of least-significant bits). Second, use these values to factor N using a second Coppersmith-style attack.

If c_1, c_2 are the b least-significant bits of d_p, d_q, then the former step has the relation

$$(N-1)x_1x_2 - (ec_2 - 1)x_1 - (ec_1 - 1)x_2 - e^2c_1c_2 + c_1 + c_2 - 1 \equiv 0 \pmod{2^b e}.$$

In practice, the Singular and Magma Gröbner basis solvers we tested were unable to efficiently handle moduli of this form. As a result of the limitations of these tools, we were unable to apply our methods to this step.

The second step, given $a = (ed_p - 1)/(p - 1)$ recovered in the first step, has relations

$$\begin{aligned} x + (ec_1 + a - 1)(2^{-b}e^{-1} \bmod aN) &\equiv 0 \pmod{ap} \\ a &\equiv 0 \pmod{a} \\ N &\equiv 0 \pmod{p}. \end{aligned}$$

For multiplicity (k_1, k_2), we can combine these relations to get shift polynomials with shared root modulo $a^{k_1}p^{k_2}$.

We ran our shift polynomial strategies on May et al.'s second step for performing Partial Key Exposure attacks on CRT-RSA. We report the results in Table 20.

Table 20: **Partial Key Exposure attacks on RSA-CRT**. Largest solvable bound for 1024-bit modulus N with 64-bit e, which is studied in [30]. Since the input relations have mixed moduli, we don't run our precomputation strategy.

	[30]			All monoms.			Graph search		
k	$\lg X_1$	Dim.	Time	$\lg X_1$	Dim.	Time	$\lg X_1$	Dim.	Time
1, 1	**60**	2	0.0	**210**	3	0.1	**210**	3	0.3
2, 1	**231**	3	0.0	**231**	3	0.2	**231**	3	0.4
4, 2	**265**	5	0.1	**265**	5	0.6	**265**	5	0.6
6, 3	**279**	7	0.1	**279**	7	1.4	**279**	7	1.5
8, 4	**287**	9	0.2	**287**	9	3.4	**287**	9	3.6
10, 5	**292**	11	0.3	**292**	11	8.8	**292**	11	9.0
12, 6	**296**	13	0.3	**296**	13	16.0	**296**	13	15.3
14, 7	**299**	15	0.5	**299**	15	28.2	**299**	15	28.8
16, 8	**301**	17	0.7	**301**	17	46.4	**301**	17	47.3
18, 9	**302**	19	0.9	**302**	≤ 20	72.9	**302**	19	78.8
20, 10	**304**	21	1.4	**304**	≤ 22	77.1	**304**	21	115.0

Our strategies match the performance of May et al.

A.11 Systems of Modular Univariate Polynomial Equations

In 2008, May and Ritzenhofen studied systems of modular univariate polynomial equations (SMUPE) [32]. This problem involves input relations with mutually coprime moduli, and the original application was polynomially related messages encrypted under separate public keys. For our experiments, we consider two messages with affine padding RSA-encrypted over two different public keys with different public exponents. This leads to the relations

$$\begin{aligned} f_1 &= (x + a_1)^3 - c_1 \equiv 0 \pmod{N_1} \\ f_2 &= (x + a_2)^5 - c_2 \equiv 0 \pmod{N_2}. \end{aligned}$$

We ran our shift polynomial strategies on SMUPE and report the results in Table 21. May and Ritzenhofen's approach only produces relations with a shared root modulo a power of $N_1^5 N_2^3$, but our ideal selection strategies can produce shift polynomials for any multiplicity.

Table 21: **SMUPE**. Largest solvable bound for affine-padded messages with 1024-bit moduli N_1, N_2 and public exponents $(3, 5)$. Since the input relations have mixed moduli, we don't run our precomputation strategy.

	[32]			All monoms.			Graph search		
k	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time
(1, 1)	**-**	-	-	**292**	8	0.4	**292**	8	0.2
(2, 1)	**-**	-	-	**369**	9	0.3	**369**	9	0.3
(2, 2)	**-**	-	-	**371**	12	0.6	**371**	12	0.6
(3, 2)	**-**	-	-	**419**	13	0.9	**419**	13	0.9
(3, 3)	**-**	-	-	**419**	15	1.4	**419**	13	1.4
(4, 3)	**-**	-	-	**443**	17	2.2	**443**	17	2.2
(4, 4)	**-**	-	-	**443**	19	3.5	**443**	17	3.6
(5, 3)	**282**	30	5.2	**461**	18	3.2	**461**	18	3.3
(5, 4)	**-**	-	-	**461**	20	5.0	**461**	18	5.0

Our strategies significantly exceed the performance of [32]. This is because our method finds novel shift polynomials, such as $\{f_1 f_2, N_1 f_2, N_2 f_1, N_1 N_2\}$, which all share a root modulo $N_1 N_2$.

A.12 Common Prime RSA

In 2014, Lu et al. studied the Common Prime RSA problem [25]. In this problem, the factors p, q of RSA modulus N have multiplicative orders which share a common prime g. That is, $p = 2ag + 1$ and $q = 2bg + 1$. The public and private exponents are e and d respectively. Lu et al. construct the following input relations, which have a shared root at $(d, p + q - 1)$:

$$\begin{aligned} N - 1 &\equiv 0 \pmod{g} \\ e - x_1 &\equiv 0 \pmod{g} \\ N - x_2 &\equiv 0 \pmod{g^2}. \end{aligned}$$

We ran our shift polynomial strategies on the Common Prime RSA problem and report the results in Table 22.

Table 22: **Common Prime RSA**. Largest solvable bound for 1000-bit modulus N with 450-bit g, which is studied in [25]. Since the input relations have mixed moduli, we don't run our precomputation strategy.

	[25]			All monoms.			Graph search			Precomputed		
k	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time
1	**132**	4	0.1	**132**	4	0.5	**132**	4	0.1	**-**	-	-
2	**207**	7	0.2	**207**	7	0.2	**207**	7	0.2	**-**	-	-
3	**234**	12	0.4	**234**	12	0.7	**234**	12	0.5	**-**	-	-
4	**257**	19	0.7	**257**	20	1.1	**257**	19	1.1	**-**	-	-
5	**272**	25	1.4	**272**	25	2.0	**272**	25	2.0	**-**	-	-
6	**285**	33	3.1	**285**	33	4.5	**285**	33	4.2	**283**	40	4.5
7	**292**	43	5.9	**292**	43	7.5	**292**	43	7.4	**-**	-	-
8	**300**	52	10.1	**300**	52	12.4	**300**	52	12.7	**-**	-	-
9	**305**	65	21.3	**305**	65	27.0	**305**	65	26.4	**-**	-	-
10	**310**	77	30.3	**310**	77	37.1	**310**	76	37.6	**-**	-	-
11	**314**	90	43.8	**314**	90	60.5	**314**	90	59.6	**-**	-	-
12	**317**	105	90.8	**317**	106	119.8	**317**	105	115.9	**317**	111	92.5

Existing bound: $\log_p X_i = 0.8100$ **Our bound:** $\log_p X_i = 0.8098$

We ran Algorithm 3 on the Common prime RSA problem with

$$\log_p X_1 = \delta, \quad \log_p X_2 = 1.1111, \quad \log_p X_{1,\text{guess}} = 0.1, \quad k_{\text{pre}} = 6,$$

$$\mathcal{M}_{\text{vert}} = \{1, x_1^7, x_2^5\}$$

and got output

$$\boldsymbol{\tau} = (0.1659, 0.2306), \quad \delta < 0.8098.$$

This nearly matches the existing asymptotic bounds of $X_1 < N^{4(0.45)^3}$ [25].

A.13 Small Secret Exponent with Multi-Power RSA

In 2014, Lu et al. studied attacks on Multi-Power RSA with small secret exponents [25]. In this problem, RSA moduli have the form $N = p^r q$ and the private exponent d is small. Lu et al.'s method is based on the observation that $ed-1$ is a multiple of p^{r-1}, and N is a multiple of p^r. For $r = 3$, this gives the relations

$$\begin{aligned} ex - 1 &\equiv 0 \pmod{p^2} \\ N &\equiv 0 \pmod{p^3}. \end{aligned}$$

We ran our shift polynomial strategies on the small secret exponent multi-power RSA problem and report the results in Table 23.

Table 23: **Small Secret Exponent with Multi-Power RSA**. Largest solvable bound for 2048-bit modulus N with $r = 3$ and 2048-bit e, which is studied in [25]. Since the input relations have mixed moduli, we don't run our precomputation strategy.

	[25]			All monoms.			Graph search			Precomputed		
k	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time	$\lg X$	Dim.	Time
10	**594**	9	0.4	**594**	9	0.8	**594**	9	0.8	**-**	-	-
20	**680**	16	1.1	**680**	16	3.5	**680**	16	3.2	**-**	-	-
30	**706**	22	4.4	**706**	22	15.5	**706**	22	15.1	**704**	21	4.1
40	**718**	29	11.9	**718**	29	47.9	**718**	29	47.9	**-**	-	-
50	**728**	35	20.8	**728**	36	124.3	**728**	36	122.5	**-**	-	-
60	**734**	42	33.5	**734**	42	279.5	**734**	42	269.5	**733**	41	33.8

Existing bound: $\log_p X_i = 1.5000$ **Our bound:** $\log_p X_i = 1.5000$

We ran Algorithm 3 on the Small Exponent Multi-Power RSA problem with

$$\log_p X = \delta, \quad \log_p X_{\text{guess}} = 0.1, \quad k_{\text{pre}} = 6, \quad \mathcal{M}_{\text{vert}} = \{1, x^4\}$$

and got output

$$\tau = 0, \quad \delta < 1.5000.$$

This matches the existing asymptotic bounds of $X < p^{3(3-1)/(3+1)}$ [25].

B Optimality of iterating Algorithm 2

In section 5.2, we introduce Algorithm 2, which finds a proper, nonempty, and suitable subset $\mathcal{S}_{\text{sub}} \subsetneq \mathcal{S}$ where $\det(\Lambda_{\mathcal{S}_{\text{sub}}})^{1/|\mathcal{S}_{\text{sub}}|} < \det(\Lambda_{\mathcal{S}})^{1/|\mathcal{S}|}$ if one exists, but there is no guarantee that $\mathcal{S}_{\text{sub}}$ minimizes $\det(\Lambda_{\mathcal{S}_{\text{sub}}})^{1/|\mathcal{S}_{\text{sub}}|}$. We iterate this algorithm until no more proper subsets are found, returning $\mathcal{S}_{\text{heur}}$. Clearly, $\mathcal{S}_{\text{heur}} \subseteq \mathcal{S}$ is a nonempty and suitable subset of $\mathcal{S}$ and $\det(\Lambda_{\mathcal{S}_{\text{heur}}})^{1/|\mathcal{S}_{\text{heur}}|} \leq \det(\Lambda_{\mathcal{S}})^{1/|\mathcal{S}|}$, but it is actually guaranteed that $\mathcal{S}_{\text{heur}}$ minimizes $\det(\Lambda_{\mathcal{S}_{\text{heur}}})^{1/|\mathcal{S}_{\text{heur}}|}$ over all possible suitable subsets of $\mathcal{S}$.

This is promising, but unfortunately the use of this algorithm remains heuristic. Although we are guaranteed to minimize the root of the lattice determinant (leading to an upper bound on the length of the shortest vector), this is not the same as minimizing the length of the shortest vector. In fact, it is possible that $\Lambda_{\mathcal{S}}$ contains a vector that is significantly shorter than the determinant bound, but that vector does not belong to the sublattice determined by $\mathcal{S}_{\text{heur}}$. This sometimes happens in practice when the shift polynomials have small coefficients, as is discussed in Section 5.3.

Nevertheless, even though the graph optimization method remains heuristic, this leads to an interesting extension of Picard's algorithm. While Picard's algorithm maximizes the *total* weight of a graph closure, our iterated variant maximizes the *average* weight of a graph closure. By setting the weights of our graph to $-\log \text{LT}(f)(\boldsymbol{X})$, this iterated process finds the closure with the largest average weight, or equivalently the smallest $\det(\Lambda_{\mathcal{S}_{\text{heur}}})^{1/|\mathcal{S}_{\text{heur}}|}$.

Theorem 3. *Algorithm 4 is correct. On input $\mathcal{G}$, $\{w_v\}_{v \in V}$, it returns a nonempty graph closure V^* that maximizes $\frac{1}{|V^*|} \sum_{v \in V^*} w_v$.*

Algorithm 4: `MaxAvgClosure`: Maximize average weight of a closure

Input : Directed nonempty graph $\mathcal{G} = (V, E)$ with weights $\{w_v\}_{v \in V}$.
Output: Nonempty graph closure $V^* \subset V$ maximizing $\frac{1}{|V^*|} \sum_{v \in V^*} w_v$

`// Modify weights so their average is 0`
1 $\hat{w}_v \leftarrow w_v - \frac{1}{|V|} \sum_{v \in V} w_v$ for all $v \in V$
2 $V_{\text{sub}} \leftarrow$ maximal closure of $\mathcal{G}$ and $\{\hat{w}_v\}_{v \in V}$ using Picard's algorithm
3 **if** $\sum_{v \in V_{sub}} \hat{w}_v = 0$ **then**
4 | **return** V
5 **else**
6 | $\mathcal{G}_{\text{sub}} \leftarrow$ directed subgraph of $\mathcal{G}$ based on closure V_{sub}
7 | **return** `MaxAvgClosure`$(\mathcal{G}_{\text{sub}}, \{w_v\}_{v \in V_{\text{sub}}})$

Proof. There are two possible outcomes in this recursive algorithm. Either we reach the base case on line 4 or we reach a recursive call on line 7. For the base case, we wish to prove that we can do no better than V. For the recursive call, we wish to show that V_{sub} is a nonempty and proper subset of V (guaranteeing termination of the algorithm) and that subgraph $\mathcal{G}_{\text{sub}}$ always contains a closure of $\mathcal{G}$ with maximum average weight. For sake of notation, let $\text{Avg}(V') = \frac{1}{|V'|} \sum_{v \in V'} w_v$.

As in the proof of Theorem 2, for nonempty V', $\sum_{v \in V'} \hat{w}_v > 0$ if and only if $\text{Avg}(V') > \text{Avg}(V)$:

$$\sum_{v \in V'} \hat{w}_v > 0 \Leftrightarrow \sum_{v \in V'} (w_v - \text{Avg}(V)) > 0 \Leftrightarrow \sum_{v \in V'} w_v > |V'| \text{Avg}(V) \Leftrightarrow$$

$$\frac{1}{|V'|} \sum_{v \in V'} w_v > \text{Avg}(V) \Leftrightarrow \text{Avg}(V') > \text{Avg}(V).$$

Let $W^* = \max \text{Avg}(V')$ over all possible nonempty closures $V' \subset V$. Since V is a closure, we have $W^* \geq \text{Avg}(V)$. There are two cases. First, we consider $W^* = \text{Avg}(V)$, so V is a valid solution (it is a nonempty graph closure that maximizes the average weight of a closure). Either V_{sub} is empty or not. If $V_{\text{sub}} = \emptyset$, then $\sum_{v \in V_{\text{sub}}} \hat{w}_v = 0$, and the algorithm returns V on line 4. If V_{sub} is nonempty, then $\text{Avg}(V_{\text{sub}}) \leq \text{Avg}(V)$ implies $\sum_{v \in V_{\text{sub}}} \hat{w}_v \leq 0$. Because Picard's algorithm returns a closure of maximum total weight, $\sum_{v \in V_{\text{sub}}} w'_v \geq \sum_{v \in V} w'_v = 0$, so $\sum_{v \in V_{\text{sub}}} w'_v = 0$, and Algorithm 4 correctly returns V.

Next, we consider the case where $W^* > \text{Avg}(V)$. Then there exists nonempty V' with $\text{Avg}(V') > \text{Avg}(V)$, so $\sum_{v \in V'} \hat{w}_v > 0$. By correctness of Picard's algorithm, we have $\sum_{v \in V_{\text{sub}}} \hat{w}_v \geq \sum_{v \in V'} \hat{w}_v > 0$, and Algorithm 4 is called recursively on line 7. We must have that $V_{\text{sub}} \neq \emptyset$ and $V_{\text{sub}} \neq V$, because otherwise $\sum_{v \in V_{\text{sub}}} \hat{w}_v = 0$, so V_{sub} is a nonempty and proper subset of V.

We know that there exists nonempty $V' \subset V$ with $\text{Avg}(V') = W^*$ that maximizes the average weight of closures of $\mathcal{G}$, but V_{sub} does not necessarily contain V'. We wish to show that there exists nonempty closure $V^* \subset V_{\text{sub}}$ with $\text{Avg}(V^*) = W^*$. If so, then the closure of $\mathcal{G}_{\text{sub}}$ with maximum average weight

which is returned by the recursive call is also a closure of $\mathcal{G}$ with maximum average weight.

We use the fact that $V' \cup V_{\text{sub}}$ and $V' \cap V_{\text{sub}}$ are both closures of $\mathcal{G}$, and we consider $V' \setminus V_{\text{sub}}$. If $\sum_{v \in V' \setminus V_{\text{sub}}} \hat{w}_v > 0$, then

$$\sum_{v \in V' \cup V_{\text{sub}}} \hat{w}_v = \sum_{v \in V' \setminus V_{\text{sub}}} \hat{w}_v + \sum_{v \in V_{\text{sub}}} \hat{w}_v > \sum_{v \in V_{\text{sub}}} \hat{w}_v.$$

This contradicts the correctness of Picard's algorithm, since V_{sub} maximizes the sum of modified weights. Thus $\sum_{v \in V' \setminus V_{\text{sub}}} \hat{w}_v \leq 0$. If $V' \setminus V_{\text{sub}} = \emptyset$, then $V' \subset V_{\text{sub}}$, so $V^* = V'$ is a nonempty closure of $\mathcal{G}_{\text{sub}}$ with $\text{Avg}(V^*) = W^*$. If $V' \cap V_{\text{sub}} = \emptyset$, then $\sum_{v \in V'} \hat{w}_v = \sum_{v \in V' \setminus V_{\text{sub}}} \hat{w}_v \leq 0$ implies $\text{Avg}(V') \leq \text{Avg}(V)$, contradicting $\text{Avg}(V') = W^* > \text{Avg}(V)$.

The only case that remains to be analyzed is therefore V' and V_{sub} with nonempty $V' \setminus V_{\text{sub}}$, nonempty $V' \cap V_{\text{sub}}$, $\sum_{v \in V'} \hat{w}_v > 0$, and $\sum_{v \in V' \setminus V_{\text{sub}}} \hat{w}_v \leq 0$. $V' \cap V_{\text{sub}}$ is a nonempty closure of $\mathcal{G}$ and $\mathcal{G}_{\text{sub}}$, and

$$\begin{aligned}
&\text{Avg}(V' \cap V_{\text{sub}}) - \text{Avg}(V') \\
=&\frac{1}{|V' \cap V_{\text{sub}}|} \sum_{v \in V' \cap V_{\text{sub}}} w_v - \frac{1}{|V'|} \sum_{v \in V'} w_v \\
=&\frac{1}{|V' \cap V_{\text{sub}}|} \sum_{v \in V' \cap V_{\text{sub}}} \hat{w}_v - \frac{1}{|V'|} \sum_{v \in V'} \hat{w}_v \\
=&\frac{1}{|V' \cap V_{\text{sub}}|} \left(\sum_{v \in V'} \hat{w}_v - \sum_{v \in V' \setminus V_{\text{sub}}} \hat{w}_v \right) - \frac{1}{|V'|} \sum_{v \in V'} \hat{w}_v \\
\geq&\frac{1}{|V' \cap V_{\text{sub}}|} \sum_{v \in V'} \hat{w}_v - \frac{1}{|V'|} \sum_{v \in V'} \hat{w}_v \\
=&\left(\frac{1}{|V' \cap V_{\text{sub}}|} - \frac{1}{|V'|} \right) \sum_{v \in V'} \hat{w}_v \\
\geq&0.
\end{aligned}$$

The last inequality is true because $|V' \cap V_{\text{sub}}| \leq |V'|$ and $\sum_{v \in V'} \hat{w}_v > 0$. Thus $\text{Avg}(V' \cap V_{\text{sub}}) \geq \text{Avg}(V') = W^*$ implies $V^* = V' \cap V_{\text{sub}}$ is a nonempty closure of $\mathcal{G}$ and $\mathcal{G}_{\text{sub}}$ achieving maximum average weight W^*.

C Discussion of Lemma 7

For completeness, we include a full proof of Lemma 7.

Proof (Lemma 7). We begin by deriving equation 3. By equation 1, a Coppersmith problem of multiplicity k with modulus p^k is heuristically solvable when

$$2^{|\mathcal{S}|(|\mathcal{S}|-1)/4} \prod_{f \in \mathcal{S}} \text{LT}(f)(X) < \left(\frac{p^k}{\sqrt{|\mathcal{S}|}} \right)^{|\mathcal{S}|+1-\ell}$$

We take the base-2 logarithm of both sides.

$$\frac{|\mathcal{S}|(|\mathcal{S}|-1)}{4} + \log_2 \prod_{f\in\mathcal{S}} \mathrm{LT}(f)(X) < (|\mathcal{S}|+1-\ell)(k\log_2 p - \log_2\sqrt{|\mathcal{S}|})$$

From Section 6.4, the log-determinant $\log_2 \prod_{f\in\mathcal{S}} \mathrm{LT}(f)(X)$ is equivalent to $\sum_{i=1}^{\ell} s_{x_i}(k,\boldsymbol{t})\log_2 X_i + \sum_{C_j\in\mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k,\boldsymbol{t})\log_2 C_j$. We substitute.

$$\frac{|\mathcal{S}|(|\mathcal{S}|-1)}{4} + \sum_{i=1}^{\ell} s_{x_i}(k,\boldsymbol{t})\log_2 X_i + \sum_{C_j\in\mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k,\boldsymbol{t})\log_2 C_j <$$

$$(|\mathcal{S}|+1-\ell)(k\log_2 p - \log_2\sqrt{|S|})$$

Next, we regroup terms.

$$\sum_{i=1}^{\ell} s_{x_i}(k,\boldsymbol{t})\log_2 X_i + \sum_{C_j\in\mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k,\boldsymbol{t})\log_2 C_j <$$

$$(|\mathcal{S}|+1-\ell)(k\log_2 p - \log_2\sqrt{|\mathcal{S}|}) - \frac{|\mathcal{S}|(|\mathcal{S}|-1)}{4}$$

We substitute $\log_2 X_i = (a_i + b_i\delta)\log_2 p$ and divide by $\log_2 p$.

$$\sum_{i=1}^{\ell} s_{x_i}(k,\boldsymbol{t})(a_i+b_i\delta) + \sum_{C_j\in\mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k,\boldsymbol{t})\log_p C_j <$$

$$(|\mathcal{S}|+1-\ell)\left(k - \frac{\log_2|\mathcal{S}|}{2\log_2 p}\right) - \frac{|\mathcal{S}|(|\mathcal{S}|-1)}{4\log_2 p}$$

We regroup to isolate δ.

$$\left(\sum_{i=1}^{\ell} a_i s_{x_i}(k,\boldsymbol{t}) + \sum_{C_j\in\mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k,\boldsymbol{t})\log_p C_j\right) + \left(\sum_{i=1}^{\ell} b_i s_{x_i}(k,\boldsymbol{t})\right)\delta <$$

$$(|\mathcal{S}|+1-\ell)\left(k - \frac{\log_2|\mathcal{S}|}{2\log_2 p}\right) - \frac{|\mathcal{S}|(|\mathcal{S}|-1)}{4\log_2 p}$$

$$\left(\sum_{i=1}^{\ell} b_i s_{x_i}(k,\boldsymbol{t})\right)\delta < (|\mathcal{S}|+1-\ell)\left(k - \frac{\log_2|\mathcal{S}|}{2\log_2 p}\right) - \frac{|\mathcal{S}|(|\mathcal{S}|-1)}{4\log_2 p} -$$

$$\left(\sum_{i=1}^{\ell} a_i s_{x_i}(k,\boldsymbol{t}) + \sum_{C_j\in\mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k,\boldsymbol{t})\log_p C_j\right)$$

$$\delta < \frac{|\mathcal{S}|k - \sum_{i=1}^{\ell} a_i s_{x_i}(k,\boldsymbol{t}) - \sum_{C_j\in\mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k,\boldsymbol{t})\log_p C_j}{\sum_{i=1}^{\ell} b_i s_{x_i}(k,\boldsymbol{t})} +$$

$$\frac{(1-\ell)k - (|\mathcal{S}| + 1 - \ell)\frac{\log_2 |\mathcal{S}|}{2\log_2 p} - \frac{|\mathcal{S}|(|\mathcal{S}|-1)}{4\log_2 p}}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, \boldsymbol{t})}$$

We set $\boldsymbol{t} = k\boldsymbol{\tau}$ and take the limit as $k \to \infty$. For any $\delta < \delta^* - \epsilon$ where $\delta^* = \lim_{k\to\infty} \ldots$, there exists k for which the Coppersmith problem is heuristically solvable.

$$\delta^* = \lim_{k\to\infty} \frac{|\mathcal{S}|k - \sum_{i=1}^{\ell} a_i s_{x_i}(k, k\boldsymbol{\tau}) - \sum_{C_j \in \mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k, k\boldsymbol{\tau}) \log_p C_j}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, k\boldsymbol{\tau})} +$$

$$\lim_{k\to\infty} \frac{(1-\ell)k - (|\mathcal{S}| + 1 - \ell)\frac{\log_2 |\mathcal{S}|}{2\log_2 p} - \frac{|\mathcal{S}|(|\mathcal{S}|-1)}{4\log_2 p}}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, k\boldsymbol{\tau})}$$

From Section 6.4, the lattice dimension $|\mathcal{S}|$ is given by the parametrization $|\mathcal{S}| = s_{dim}(k, k\boldsymbol{\tau})$. We substitute.

$$\delta^* = \lim_{k\to\infty} \frac{k s_{dim}(k, k\boldsymbol{\tau}) - \sum_{i=1}^{\ell} a_i s_{x_i}(k, k\boldsymbol{\tau}) - \sum_{C_j \in \mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k, k\boldsymbol{\tau}) \log_p C_j}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, k\boldsymbol{\tau})} +$$

$$\lim_{k\to\infty} \frac{(1-\ell)k - (s_{dim}(k, k\boldsymbol{\tau}) + 1 - \ell)\frac{\log_2 s_{dim}(k,k\boldsymbol{\tau})}{2\log_2 p} - \frac{s_{dim}(k,k\boldsymbol{\tau})(s_{dim}(k,k\boldsymbol{\tau})-1)}{4\log_2 p}}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, k\boldsymbol{\tau})}$$

By Heuristic 2, $s_{dim}(k, k\boldsymbol{\tau})$ is a polynomial in k. We eliminate low-degree terms from the right-hand expression which do not contribute to the value of the limit.

$$\delta^* = \lim_{k\to\infty} \frac{k s_{dim}(k, k\boldsymbol{\tau}) - \sum_{i=1}^{\ell} a_i s_{x_i}(k, k\boldsymbol{\tau}) - \sum_{C_j \in \mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k, k\boldsymbol{\tau}) \log_p C_j}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, k\boldsymbol{\tau})} +$$

$$\lim_{k\to\infty} \frac{-1}{4\log_2 p} \frac{s_{dim}(k, k\boldsymbol{\tau})^2}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, k\boldsymbol{\tau})}$$

Our requirement on $\log p$ in the statement of Lemma 7 means the second limit is zero. Thus the Coppersmith problem is heuristically solvable for $\delta < \delta^* - \epsilon$ where

$$\delta^* = \lim_{k\to\infty} \frac{k s_{dim}(k, k\boldsymbol{\tau}) - \sum_{i=1}^{\ell} a_i s_{x_i}(k, k\boldsymbol{\tau}) - \sum_{C_j \in \mathrm{LC}(\mathcal{S}_1)} s_{C_j}(k, k\boldsymbol{\tau}) \log_p C_j}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, k\boldsymbol{\tau})}$$

To prove the claim about running time, observe that δ^* is the limit of a rational function. For a given $\epsilon > 0$, the limit converges polynomially quickly, and there exists $k = \mathrm{poly}(\epsilon^{-1})$ such that the use of multiplicity k heuristically solves the multivariate Coppersmith problem for bound $\delta < \delta^* - \epsilon$. By this choice of multiplicity, we also have $\log p$ lower bounded by $\mathrm{poly}(k) = \mathrm{poly}(\epsilon^{-1})$. Using the precomputation method involves substituting actual coefficients into the precomputed shift polynomial set $\mathcal{S}_1$, computing $\mathcal{S}_{k,\boldsymbol{t},\mathrm{ul}}$, constructing the dual lattice, reducing the dual lattice, and performing the final root recovery

step on the reduced vectors. Substitution into $\mathcal{S}_1$ only takes time $\text{poly}(\log_p)$ and results in a set with a constant number of coefficients of size $\text{poly}(\log_p)$.

When computing $\mathcal{S}_{k,\boldsymbol{t},\text{ul}}$, observe that Lemma 6 guarantees that $|\mathcal{S}_k|$ is polynomial in k, so throughout the computation of $\mathcal{S}_k$, all sets have at most $\text{poly}(k)$ elements, each element is the product of at most k polynomials in the substituted $\mathcal{S}_1$, and so the number of coefficients in each polynomial in $\mathcal{S}_k$ is $\text{poly}(k)$ and the size of each coefficient is $\text{poly}(k, \log p)$. Computing $\mathcal{S}_{k,\boldsymbol{t}}$ from $\mathcal{S}_k$ is also polynomially bounded, and so is computing the normal form to arrive at $\mathcal{S}_{k,\boldsymbol{t},\text{ul}}$.

Given the shift polynomials in $\mathcal{S}_{k,\boldsymbol{t},\text{ul}}$ and the bounds $\boldsymbol{X}$, we can build the Howgrave-Graham dual lattice. Every coefficient in the shift polynomials has bit length $\text{poly}(k, \log p)$. The largest degree of any monomial in $\mathcal{S}_{k,\boldsymbol{t},\text{ul}}$ is $\text{poly}(k)$, so the scaling factors $\prod_{i=1}^{\ell} X_i^{e_i}$ (where $\prod_{i=1}^{\ell} x_i^{e_i}$ is a monomial) have bit length $\text{poly}(k, \log p)$ as well. Therefore the dual lattice has dimension $|\mathcal{S}_{k,\boldsymbol{t},\text{ul}}| = \text{poly}(k)$ and entries of size $\text{poly}(k, \log_p)$. Reducing this lattice takes time $\text{poly}(k, \log_p)$.

Excluding the final root recovery step, this means that the precomputation strategy takes time $\text{poly}(k, \log_p)$, and by choosing $\log p = \Theta(\text{poly}(k)$, the overall running time is polynomial in ϵ^{-1}.

C.1 Example application

To demonstrate how Lemma 7 can be used, we apply it to a relatively simple example. Note that this example is an instance of the "Generalized Rectangle" from Jochemsz and May [22, Appendix A]. They predict that their basic strategy finds a small modular root when $\prod_{i=1}^{\ell} X_i^{\lambda_i} < p^{2/((\ell+1)D)}$. In this worked example, $\ell = 2$, $\lambda_i = 1$, $X_1 = X_2$, and $D = 1$, so their strategy leads to the bound $X_i < p^{1/3}$. Our proof, which uses our own lemma, arrives at the same bound.

Corollary 1. *We examine the multivariate Coppersmith problem in $\ell = 2$ variables with known modulus p and relation*

$$f(x_1, x_2) = x_1 x_2 + c_1 x_1 + c_2 x_2 + c_3.$$

We wish to find bounded root (r_1, r_2) with $f(r_1, r_2) \equiv 0 \pmod{p}$ where $|r_i| < p^{\delta}$. Assume Heuristics 1 and 2 hold. For any choice of $\epsilon > 0$, the Coppersmith problem is solvable for $\delta < 1/3 - \epsilon$.

Proof. We choose the precomputed set $\mathcal{S}_1 = \{f, px_1, px_2, p\}$, noting that the corresponding polytope is a square. For ease of explanation, we pick $\boldsymbol{\tau} = 0$ so $\boldsymbol{t} = 0$ always. However, we could have also left it unspecified and explicitly construct $\mathcal{S}_{k,\boldsymbol{t},\text{ul}}$ for different values of $\boldsymbol{t}$ with entries between 0 and 2. Then we would perform multivariate polynomial interpolation. However, with $\boldsymbol{\tau} = 0$, observe that $\hat{J}_{\infty} = \langle 0 \rangle$, so $\mathcal{S}_{k,\boldsymbol{t},\text{ul}} = \mathcal{S}_k$. We may explicitly calculate the dimension and determinant of the lattice for $\mathcal{S}_k$ at different multiplicities k. For example, $\mathcal{S}_1$ has 4 shift polynomials, and the product of leading terms is $x_1^2 x_2^2 p^3$. We calculate $\mathcal{S}_2 = \{f^2, px_1 f, px_2 f, pf, p^2 x_1^2, p^2 x_2^2, p^2 x_1, p^2 x_2, p^2\}$, seeing it has 9 shift

polynomials, and the product of leading monomials is $x_1^9 x_2^9 p^{13}$. Continuing the calculation,

$$\begin{array}{lll}
k=1 & \dim \Lambda_{\mathcal{S}_1} = 4 & \log\det \Lambda_{\mathcal{S}_1} = 2X_1 + 2X_2 + 3\log p \\
k=2 & \dim \Lambda_{\mathcal{S}_1} = 9 & \log\det \Lambda_{\mathcal{S}_1} = 9X_1 + 9X_2 + 13\log p \\
k=3 & \dim \Lambda_{\mathcal{S}_1} = 16 & \log\det \Lambda_{\mathcal{S}_1} = 24X_1 + 24X_2 + 34\log p \\
k=4 & \dim \Lambda_{\mathcal{S}_1} = 25 & \log\det \Lambda_{\mathcal{S}_1} = 50X_1 + 50X_2 + 70\log p \\
k=5 & \dim \Lambda_{\mathcal{S}_1} = 36 & \log\det \Lambda_{\mathcal{S}_1} = 90X_1 + 90X_2 + 125\log p.
\end{array}$$

By Heuristic 2, we conclude

$$\begin{aligned}
s_{\dim}(k,0) &= k^2 + 2k + 1 \\
s_{x_1}(k,0) = s_{x_2}(k,0) &= k^3/2 + k^2 + k/2 \\
s_p(k,0) &= 2k^3/3 + 3k^2/2 + 5k/6.
\end{aligned}$$

By Lemma 7, noting $a_i = 0$ and $b_i = 1$, we see that the Coppersmith problem is solvable for $\delta < \delta^* - \epsilon$ where $\delta^* =$

$$\lim_{k\to\infty} \frac{k s_{\dim}(k,0) - \sum_{C_j \in \{1,p\}} s_{C_j}(k,0)\log_p C_j}{\sum_{i=1}^{\ell} s_{x_i}(k,0)}.$$

Substituting in $s_{\dim}$, s_{x_1}, s_{x_2}, and s_p (excluding low-order terms which do not affect the limit), we see

$$\delta^* = \lim_{k\to\infty} \frac{kk^2 - \frac{2}{3}k^3}{2(\frac{1}{2}k^2)} = \frac{1}{3},$$

implying the bound $X_1, X_2 < p^{1/3}$.