Solving Multivariate Coppersmith Problems with Known Moduli

Keegan Ryan

University of California, San Diego, 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 signicant 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 four of the problems. In four problems, we find smaller and more efficient lattice constructions, and in two 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 signicant 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 [\[13\]](#page-28-0), Coppersmith realized that the coefficients of shifted polynomial xf induce an independent linear constraint, while only introducing a single new monomial. By considering shift polynomials of the form x^if^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 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. 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,](#page-2-0) we give general background on multivariate Coppersmith problems, and in Section [3,](#page-6-0) 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,](#page-8-0) we give a simple and novel algorithm to select shift polynomials from the ideal that are defined over a fixed set of monomials. This algorithm is based on the theory of Gröbner bases over Euclidean domains. We prove that our algorithm is optimal in two parts: we show that our definition of optimality is meaningful and unique, and we show that our algorithm achieves this notion

of optimality. Finally, we compare our new algorithm to the algorithm in [\[33\]](#page-29-0) and show how ours finds significantly better shift polynomials than their design, even though their algorithm claimed to be optimal.

In Section [5,](#page-13-0) we show how to select the set of monomials. We describe a set of monomials \mathcal{M}_{big} and use lattice reduction theory to show that this set is large enough to guarantee that all suitably small vectors exist in the resulting Coppersmith lattice. The main drawback of this result is that the resulting lattice may have large rank, increasing the cost of lattice reduction. To find a better subset of \mathcal{M}_{big} , we describe an algorithm that automatically searches the lattice for dense sublattices. Our algorithm does not rely on lattice reduction and is based on an optimization technique for weighted directed graphs. Our algorithm is not guaranteed to find a sublattice if one exists, but if it finds a sublattice. it is guaranteed to contain short vectors. In practice, this algorithm is fast and highly effective.

In Section [6,](#page-18-0) we show how the expensive Gröbner basis computations in Section [4](#page-8-0) can be avoided when working with powers of ideals. We describe a simplied strategy that is similar to prior work which takes a small but good set of shift polynomials S in an ideal J and outputs a larger set of shift polynomials \mathcal{S}_k in ideal J^k . We use the theory of Ehrhart polynomials to analyze the asymptotic behavior of this simplified approach. We conclude this section with a procedure to automatically prove asymptotic bounds for any multivariate Coppersmith problem, supposing standard heuristic assumptions are true. While this procedure requires some hand-selected inputs, it drastically simplifies the existing process of proving asymptotic bounds, and it agrees with existing bounds for well-studied classes of multivariate Coppersmith problems.

Finally, in Section [7](#page-23-0) 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 the most advanced shift polynomial strategies previously described, including Herrmann and May's unravelled linearization [\[19\]](#page-28-1), the exponent tricks of Lu, Zhang, Peng, and Lin [\[25\]](#page-29-1), and the linear algebra strategy of Xu, Sarkar, Hu, Huang, and Peng [\[46\]](#page-30-0). 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 $m = \prod_{1}^{\ell} x_i^{e_i}$ multiplied by coefficients c_m . For a particular *monomial ordering*, each nonzero polynomial f has a leading term $LT(f)$ with leading monomial $LM(f)$ and leading coefficient $LC(f)$. For brevity, we often use vector notation: $f(x)$ is shorthand for $f(x_1, \ldots, x_\ell)$. We are also sometimes casual with function notation: for example, $LM(S)$ means the set of leading monomials of polynomials in S .

2.1 Multivariate Coppersmith problems

Definition 1 (Multivariate Coppersmith problem). A multivariate Cop $persmith\ problem\ in\ \ell\ variables\ involves\ finding\ one\ or\ more\ bounded,\ shared$ roots to a system of modular or integer polynomials. For bounds $\mathbf{X} \in \mathbb{Z}^{\ell}$ and constrained input polynomials $\mathcal{F} \subset \mathbb{Z}[\bm{x}]$, find a small root $\bm{r} \in \mathbb{Z}^{\ell}$ with $|r_i| < X_i$ for $i = 1, \ldots, \ell$ such that r satisfies the constraint of polynomial $f_i \in \mathcal{F}$. This constraint is either modular or integer:

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

where the moduli p_i may either be known or bounded below by a known value.

This definition includes modular Coppersmith problems, integer Coppersmith problems, and Coppersmith problems with mixed moduli, but our focus in this work is multivariate problems with at least one modular constraint and where multiples N_i of moduli p_i are known. This includes many interesting applications of Coppersmith's method. Implementations of this method almost all follow the same general outline:

- 1. Combine polynomials in $\mathcal F$ to generate a set of constrained *shift polynomials*.
- 2. Construct a lattice basis using the shift polynomials.
- 3. Run a lattice reduction algorithm to obtain a reduced basis.
- 4. Interpret short vectors in the lattice in a way that reveals root r .

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 [\[13\]](#page-28-0). All shift polynomials are of the form x^if^j , and Coppersmith showed that lattice reduction succeeds for this selection of shift polynomials when $\log_n X \leq 1/\deg f$, which is provably optimal [\[11\]](#page-28-2). May gave a generalization of Coppersmith's result where p is unknown, but a multiple N of p is known [\[28\]](#page-29-2). Coppersmith observed that the same shift polynomials and lattice methods may work in principal 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) [\[22\]](#page-28-3). They describe a shift polynomial strategy based on the monomial sets constructed by considering monomials in $x^t f^k$. In Appendix A, they show how their generalized and heuristic strategy reproduces the same bounds for problems studied by Boneh and Durfee [\[7\]](#page-27-0) and Blömer and May [\[5\]](#page-27-1).

More recently, Meers and Nowakowski studied systems of multiple multivariate polynomials with known modulus p [\[33\]](#page-29-0). They describe a heuristic strategy for choosing a set of monomials and give an algorithm for selecting shift polynomials from the set of monomials. The algorithm is based on searching combinations of input polynomials for which the leading monomial of the product is in the monomial set. They claim 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 problemspecific strategies. As more intricate shift polynomial strategies are developed and analyzed, the bounds on recoverable roots slowly increase. Take for example the Modular Inversion Hidden Number Problem (MIHNP) from Boneh et al. in 2001 [\[8\]](#page-27-2) which studies ℓ polynomials of the form $f_i(\alpha, x) = \alpha x_i + c_{i1}\alpha + c_{i2}x_i + c_{i3}$ modulo 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 [\[44\]](#page-30-1) and 2018 [\[46\]](#page-30-0), culminating the breakthrough result by Xu et al. at Crypto 2019 [\[47\]](#page-30-2) that $\log_p X_i < \ell/(\ell+1)$ is heuristically and asymptotically solvable. The results were refined further in 2023 [\[48\]](#page-30-3).

Similar incremental improvements in shift polynomial strategies appear for the Elliptic Curve Hidden Number Problem [\[45](#page-30-4)[,49\]](#page-30-5) and RSA-CRT with small private exponents [\[26,](#page-29-3)[4,](#page-27-3)[23](#page-28-4)[,20,](#page-28-5)[43\]](#page-30-6), not to mention the many RSA partial key exposure variants [\[16](#page-28-6)[,1,](#page-27-4)[40](#page-30-7)[,42](#page-30-8)[,30,](#page-29-4)[31\]](#page-29-5). 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 M , there are a couple of ways to build the lattice. Coppersmith's original work [\[13\]](#page-28-0) builds an extended lattice basis of dimension $|\mathcal{M}| + |\mathcal{S}|$ and finds a projected sublattice Λ of dimension $|\mathcal{M}|$. Each coordinate in Λ corresponds to a monomial in M , and it is equivalent (up to scaling) to the following:

$$
\left\{ \left(\frac{a_m}{m(\mathbf{X})} \right)_{m \in \mathcal{M}} \Big| \sum_{m \in \mathcal{M}} c_m a_m \text{ satisfies the constraint of } \sum_{m \in \mathcal{M}} c_m m \in \mathcal{S} \right\}.
$$

That is, the lattice vectors represent solutions to a linearized version of S . This is the primal lattice.

Howgrave-Graham gave an alternative consruction [\[21\]](#page-28-7). If all $f \in \mathcal{S}$ satisfy the same constraint (such as a shared 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 $S = \{N, Nx_1, Nx_2, x_1x_2 + ax_1 +$ $b, x_1x_2^2 + ax_1x_2 + bx_2$ is spanned by the rows of the basis matrix

The vectors in the lattice correspond to (scaled coefficient vectors of) polynomials. When 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

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

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. 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

 \ddotsc

Given a lattice Λ of rank d, a lattice reduction algorithm outputs a reduced basis B' of Λ consisting of short, nearly orthogonal vectors. The LLL algorithm [\[24\]](#page-29-6) runs in polynomial time, and the basis vectors in the output satisfy

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

where $\lambda_i(\varLambda)$ is the i^{th} minimum of the lattice, or the minimum radius of a ball at the origin containing i linearly independent vectors. These bounds are found in [\[27\]](#page-29-7) and [\[35\]](#page-29-8), and analogous bounds can be derived for more modern reduction algorithms capable of reducing lattices of large rank [\[39\]](#page-29-9).

2.5 Root recovery

In the primal lattice, the existence of a bounded root $\bm{r} \in \mathbb{Z}^{\ell}$ implies the existence of a short vector $\left(\frac{m(\mathbf{r})}{m(\mathbf{X})}\right)$ $\frac{m(\mathbf{r})}{m(\mathbf{X})}$ with max norm ≤ 1 . We hope that this vector is uniquely short and found by lattice reduction, so r can be recovered.

In the dual lattice, a short vector corresponds to a polynomial g with small coefficients. *q* is an integer linear combination of polynomials in S , so it satisfies the same constraint (for example $g(r) \equiv 0 \pmod{p}$). The following result of Håstad [\[18\]](#page-28-8) and Howgrave-Graham [\[21\]](#page-28-7) is used to show that $g(r) = 0$ over the integers, not just modulo p. We refer to this as the HHG bound.^{[1](#page-5-0)}

 1 This is often referred to as the Howgrave-Graham bound, but as May notes [\[29\]](#page-29-10), it appears in Håstad's earlier work as well.

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

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 $\mathbb{Q}[x]$ of polynomials recovered by lattice reduction is zero-dimensional.

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

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

It is common to see this in an asymptotic form, where we consider arbitrarily large p, which scales independently of $|\mathcal{S}|$, and $|\mathcal{S}|$ scales independently of ℓ .

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

2.6 On the possibility of a non-heuristic algorithm

In [\[14\]](#page-28-9), Coppersmith showed that an efficient provable approach to solving multivariate Coppersmith problems is not possible. More accurately, he shows that a method to find small solutions to modular equation $ax_1^2 + bx_2 - c \equiv 0 \pmod{p}$ implies an efficient solution to a 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 practical implementations are quite ef ficient. In line with previous work, our focus is also on developing a 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 $\mathbb{Z}[\mathbf{x}]$. Prior work has also shown the connection between shift polynomials and ideals [\[2\]](#page-27-5), but we hope to elaborate on this in greater depth.

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

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

This recalls 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 dened for ideals:

$$
J + J' = \{ f + f' \mid f \in J, f' \in J' \} \quad J \times J' = \{ \sum_i f_i f'_i \mid f_i \in J, f'_i \in J' \}.
$$

Multiplication recalls 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 [\[2\]](#page-27-5). 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 $J \times J'$ 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 encompases all shift polynomial strategies we are aware of, including the linear algebra-based strategy in $[46,47,48]$ $[46,47,48]$ $[46,47,48]$, the exponent tricks in [\[25\]](#page-29-1) or the technique of unravelled linearization [\[19\]](#page-28-1).

3.1 Unravelled linearization

In 2009, Herrmann and May proposed a novel technique for shift polynomial selection called unravelled linearization [\[19\]](#page-28-1). In essence, they observed that in the input relation $f(x) = x_1^2 - x_2 + ax_1 + b \equiv 0 \pmod{p}$, it helps to group together ("linearize") the terms $(x_1^2 - x_2) \mapsto u$ into a new bounded variable, so $g(x, u) = u + ax_1 + b \equiv 0 \pmod{p}$. 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 $f_{ul}(x_1, x_2, u) = x_1^2 - u - x_2$. Observe that $g \in \langle f, p, f_{ul} \rangle \subset \mathbb{Z}[x, u]$ and g has a small root modulo p bounded by $(X_1, X_2, X_1^2 + X_2)$. Furthermore, $p^{k-i}g_{i,j} \in \langle f, p, f_{\text{ul}} \rangle^k$, and the unravelling of this polynomial is in $\langle f, p \rangle^k + \langle f_{\text{ul}} \rangle$. If we have a monomial ordering where $u < x_1^2$, then reduction by $\langle f_{\rm ul} \rangle$ 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 $f_{ul} \in \mathcal{F}'$ with an integer constraint.

3.2 Determining the shift polynomial ideal

For constrained input polynomials $\mathcal F$ and desired root r , define

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

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 $f \in J_k$, $f(r) \equiv 0 \pmod{p^k}$.

However, we also consider situations where the relations involve multiple moduli. This includes moduli that are distinct RSA semiprimes [\[32\]](#page-29-11) or different powers of an unknown divisor p [\[25\]](#page-29-1). 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 $\boldsymbol{k}\in\mathbb{Z}_{\geq 0}^{|Q|}$ $\geq 0^{\mathsf{Q}}$, we define

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

Observe that for all $e \in E$, $f \in \prod_i \hat{J}_{p_i}^{e_i}$ satisfies $f(r) \equiv 0 \pmod{\prod_i p_i^{e_i}}$, so $f \in J_{\mathbf{k}}$ satisfies $f(r) \equiv 0 \pmod{\prod_j q_j^{k_j}}$. One can efficiently compute $J_{\boldsymbol{k}}$ from smaller multiplicities $J_{k'}$ 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 [\[22\]](#page-28-3), and more recently by Meers and Nowakowski in 2023 [\[33\]](#page-29-0). 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 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 pricipal ideal domains and Euclidean domains. For our purposes, this is used to find Gröbner bases for ideals in $\mathbb{Z}[\mathbf{x}]$. While there are some crucial differences compared to the more familiar case of ideals in $\mathbb{Q}[x]$, 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 setting. As discussed in section [2.5,](#page-5-1) Gröbner bases for ideals in $\mathbb{Q}[x]$ are frequently used to find the small root following lattice reduction. Some more recent works replace this ring with $\mathbb{F}_p[x]$ and use the Chinese remainder theorem to reconstruct the zero-dimensional variety [\[33](#page-29-0)[,48](#page-30-3)[,49\]](#page-30-5). Herrmann and May also used Gröbner bases in 2010 to find a nontrivial relationship between unravelled linearization variables [\[20\]](#page-28-5).

Gröbner bases are defined relative to a monomial ordering. There are many different valid orderings, but we typically use the weight order defined by the bounds on the unknown variables. If small roots are bounded by $|r| < X$, then we order monomials by $\langle \mathbf{x}, \mathbf{w} \rangle$ where $m_1 \langle m_2 \rangle$ if $m_1(\mathbf{X}) \langle m_2(\mathbf{X}) \rangle$. In case of ties, we fall back to lexicographic order.^{[2](#page-9-0)}

The textbook of Becker and Weispfenning [\[3\]](#page-27-6) describes an algorithm to find what they call a D-Gröbner basis G of an ideal $J \in R[\mathbf{x}]$ 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,](#page-27-6) Exercise 10.5].

Lemma 2 (Becker and Weispfenning). Let $J \in R[x]$ be an ideal, and let G be a D-Gröbner basis of J. Every nonzero $f \in J$ is top-D-reducible modulo G. That is, there exists $g \in G$ where $LT(g)|LT(f)$.

Further, we use the fact that if R is a Euclidean domain, a polynomial f has a unique normal form f' with respect to Gröbner basis G [\[3,](#page-27-6) Theorem 10.23]. $f - f' \in \langle G \rangle$ and all terms in f' are irreducible by G .

To illustrate the difference between Gröbner bases 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 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 was able to implicitly do 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

Our method for finding the optimal shift polynomials within an ideal is given in Algorithm [1.](#page-10-0) We call attention to two important steps in this algorithm. First, in line [1](#page-10-1) we compute the finite set of monomials $\overline{\mathcal{M}}$ with the property that if

 2 We take advantage of this tie-breaking when using unravelled linearization. If we are using the unravelled linearization $u = x_1^2 - x_2$, as in Section [3.1,](#page-7-0) and set the weights to $(\log X_1, \log X_2, 2 \log X_1)$, then the tie-breaking of lexicographic order leads to $u < x_1^2$ as desired.

Algorithm 1: OptimalShiftPolys

Input : Ideal J, monomial set M , monomial ordering \lt Output: Shift polynomials S $1 \overline{\mathcal{M}} \leftarrow$ finite superset of $\mathcal M$ s.t. all monomials appearing in $\overline{\mathcal{S}}$ belong to $\overline{\mathcal{M}}$. 2 $G \leftarrow$ D-Gröbner basis of $(J, <)$ $3\overline{\mathcal{S}}\leftarrow \{\}$ 4 for $m \in \overline{\mathcal{M}}$ do 5 $\mid T \leftarrow \{g \in G \mid LM(g) \text{ divides } m\}$ 6 if $T \neq \emptyset$ then $\tau \mid \quad \mid \quad g \leftarrow \text{argmin}_{g \in T} |\text{LC}(g)|$ 8 $h \leftarrow g \frac{m}{\text{LM}(g)}$ // Ensure $\text{LM}(h) = m$ $9 \mid h' \leftarrow \text{LT}(h) + \text{normal_form}_G(h - \text{LT}(h))$ 10 $\overline{S} \leftarrow \overline{S} \cup \{h'\}$ 11 Find the linear subspace of span(\overline{S}) where all coefficients of $m \in \overline{\mathcal{M}} \setminus \mathcal{M}$ are 0 12 $S \leftarrow$ basis of this subspace 13 return S

f is in normal form and $LM(f) \in \overline{\mathcal{M}}$, then all monomials in f are in $\overline{\mathcal{M}}$. This seems complicated, but if \lt is a weighted monomial ordering, then $\overline{\mathcal{M}}$ can simply be the finite set of monomials \leq max_{$\lt M$}. Second, the operation on line [11](#page-10-2) is done by interpreting span(\overline{S}) as a vector space of coefficients over $\mathbb Z$ and using standard row operations on the basis to zero out desired coefficients. In practice, both of these steps are often skipped because $\mathcal{M} = \overline{\mathcal{M}}$, but we include them here for completeness.

Additionally, the computation of the normal form on line [9](#page-10-3) is not strictly necessary for this section, but it is important for Section [5.](#page-13-0) In the context of our full approach to solving Coppersmith-style problems, this normalization step also has the effect of ensuring the dual lattice basis is size-reduced before lattice reduction.

4.3 Optimality of the algorithm

We claim that Algorithm [1](#page-10-0) is optimal for a given choice of monomials, but we must define our notion of optimality. In essence, the selection of shift polynomials must perform at least as well as any other choice. Specifically, if the process of building a basis for the dual lattice, reducing the basis, and finding suitably short vectors is guaranteed to succeed for one choice of shift polynomials, it should succeed for the optimal choice of shift polynomials as well. We use a combination of properties of both lattices and 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 M , the dual lattice construction specifies a natural embedding $\varphi: J \to \mathbb{Z}^{|\mathcal{M}|}$ that converts shift polynomials to scaled integer coefficient vectors. When the nonzero terms of f only involve monomials in M (f is "defined over" M), the mapping φ is invertible. Furthermore, φ is additively homomorphic: $\varphi(f+g) = \varphi(f) + \varphi(g)$, so adding together polynomials in the ideal corresponds to adding together vectors in a lattice. When the shift polynomials are defined over M , the dual lattice $\Lambda_{\mathcal{S}}$ is the span of vectors $\{\varphi(f) \mid f \in S\}.$

If we consider two sets of shift polynomials $(S_1 \text{ and } S_2)$, the set-union of shift polynomial sets corresponds to the lattice-union of dual lattices:

$$
\Lambda_{\mathcal{S}_1\cup\mathcal{S}_2}=\Lambda_{\mathcal{S}_1}\cup\Lambda_{\mathcal{S}_2}.
$$

Thus $\Lambda_{\mathcal{S}_1}$ is a sublattice of $\Lambda_{\mathcal{S}_1\cup\mathcal{S}_2}$, informing the following definition.

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

Lemma 3. The optimal dual lattice is unique.

Proof. Consider two optimal lattices Λ_{S_1} and Λ_{S_2} . $S_1 \cup S_2 \subset J$ is defined over M, so $\Lambda_{S_1} \subset \Lambda_{S_1 \cup S_2} \subset \Lambda_{S_1}$ by lattice union and optimality of Λ_{S_1} . Thus $\Lambda_{S_1} =$ $\Lambda_{\mathcal{S}_1\cup\mathcal{S}_2}$. The same is true of $\Lambda_{\mathcal{S}_2}$, so therefore $\Lambda_{\mathcal{S}_1}=\Lambda_{\mathcal{S}_2}$.

The definition of optimality based on sublattices is also useful for bounding the length of vectors found by lattice reduction. We recall from section [2.4](#page-5-2) that reduced basis vectors can be bounded by the successive minima λ_i of a lattice.

Lemma 4. For all S defined over M, the optimal dual lattice Λ^* for M 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 lenghs are obtained by considering the optimal lattice. Finally, we arrive at the main result of this section.

Theorem [1](#page-10-0). Let S be the shift polynomials returned by Algorithm 1 for ideal *J*, monomial set M and monomial ordering \lt . Then $\{\varphi(f) \mid f \in \mathcal{S}\}\$ is a basis for the optimal lattice.

Proof. The elements of S are Z-linearly independent, so if Λ_S is optimal, then 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 v . By the homomorphic property of φ , if f is in the integer linear span of S, then $v \in A_{\mathcal{S}}$, proving $A_{\mathcal{S}} \subset A_{\mathcal{S}}$.

f is defined over M, so $f \in \text{span}(\mathcal{S})$ if $f \in \text{span}(\overline{\mathcal{S}})$. We will iteratively subtract integer multiples of elements of \overline{S} until $f = 0$. Since $\overline{S} \subset J, f \in J$ after each subtraction. First, if $f = 0$, then we are done. If $f \neq 0$, then let $m = LM(f)$ and let T, g, h, h' be the corresponding values in Algorithm [1.](#page-10-0) T is nonempty by Lemma [2.](#page-9-1)

Since $LM(h') = m$ and $h' \in \overline{S}$, subtract an integer multiple of h' from f so the coefficient of m in f is in $\{0, 1, \ldots, |LC(h')|-1\}$. Note that $LC(h') = LC(g)$. Assume that that this coefficient is nonzero. Since $f \in J$, Lemma [2](#page-9-1) shows there exists $\hat{g} \in G$ where $LT(\hat{g})|LT(f)$. $LM(\hat{g})|LM(f)$ and $LM(f) = m$, so $\hat{g} \in T$, but $LC(\hat{g})|LC(f) \Rightarrow |LC(\hat{g})| \leq |LC(g)| - 1$, contradicting the minimality of $|LC(g)|$ 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 [\[33\]](#page-29-0). To solve the Commutative Isogeny Hidden Number Problem for CSURF key exchange, they consider a system of two polynomials modulo prime N with known c .

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

\n
$$
g = (c_3 + x_3)^2 + 12(c_3 + x_3) - 4(c_3 + x_3)(c_1 + x_1)^2 - 8(c_1 + x_1)^2 + 36
$$

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 by considering products of $LM(f)$ and $LM(q)$ that multiply to elements in M. However, their algorithm is sensitive to the choice of monomial order. Using lexicographic order $x_3 < x_2 < x_1$ results in a lattice with determinant $X_1^{49}X_2^{30}X_3^{27}N^{54}$, but lexicographic order $x_2 < x_1 < x_3$ results in improved determinant proportional to N^{52} . Degree lexicographic order gives N^{53} . Their definition of optimality implicitly requires fixing a monomial order, and choosing incorrectly leads to worse performance in practice.

Our definition of optimality is independent of monomial order. Algorithm [1](#page-10-0) finds the optimal lattice for shift polynomial ideal $\langle N, f, g \rangle^2$ has determinant proportional to N^{51} ; this improvement is possible because the Gröbner basis calculation finds nontrivial polynomial combinations of f and g . For example, degree lexicographic order gives $LM(f) = x_1 x_2^2$ and $LM(g) = x_1^2 x_3$. For monomial $m = x_2^2 x_3^2$, Algorithm 2 in [\[33\]](#page-29-0) checks that neither LM(f) nor LM(g) divide m, and it returns the "optimal" shift polynomial $N^2 x_2^2 x_3^3$. Using the Gröbner basis, we find a shift polynomial with leading term $Nx_2^2x_3^3$:

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

This shift polynomial decreases the lattice determinant and contributes to the optimal lattice construction.

5 Monomial selection

Section [4](#page-8-0) describes how to select shift polynomials within an ideal based on a set M of monomials, but it does not explain how to pick M . This matters, because if M is too small, suitably short vectors may not exist in the optimal dual lattice, and if M is too large, LLL reduction may not find the short vectors that do exist. In this section, we describe a monomial set \mathcal{M}_{big} that is provably large enough, and we describe a heuristic method to find a small subset $\mathcal{M}_{sub} \subset \mathcal{M}_{big}$ with useful bounds on the shortest vector.

Our methods assume that multiple N of modulus p is known. We have $p \leq N$, and in some cases, an even tighter upper bound P on p may be known. This setting has the nice property that Algorithm [1](#page-10-0) outputs shift polynomials S that are $(\mathcal{M}, <)$ -suitable.

Definition 3 ([\[33\]](#page-29-0)). Given monomial set M and monomial ordering \lt , a set of shift polynomials S is (\mathcal{M}, \leq) -suitable if every $f \in \mathcal{S}$ is defined over M, and for each $m \in \mathcal{M}$, there is a unique $f \in \mathcal{S}$ with $LM(f) = m$.

Since $N \equiv 0 \pmod{p}$, $N \in J$ and Lemma [2](#page-9-1) guarantees that the D-Gröbner basis of J includes a polynomial with leading monomial [1](#page-10-0). Thus T in Algorithm 1 is always nonempty, and \overline{S} is by construction $(\overline{\mathcal{M}}, \langle \rangle)$ -suitable. The corresponding span of \overline{S} is full rank, so the projection of the linear subspace is also full rank, and there exists a triangular basis S of this subspace that is (\mathcal{M}, \leq) -suitable.

5.1 Provably good monomial selection

 $(M, <)$ -suitability allows us to construct triangular lattice bases, and the Gram-Schmit norm $||b^*||$ of a particular row in the basis is given by diagonal element $LT(f)(X)$ of the corresponding $f \in S$. This property allows us to prove the following.

Lemma 5. Let M be any monomial set. We are given bounds X and ideal J in which all polynomials share a root modulo p. An upper bound $P > p$ is known. $Define$

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

Let A be the optimal dual lattice of M and J. If $v \in A$ satisfies the bound of H åstad/Howgrave-Graham, then the corresponding polynomial is defined over $\mathcal{M} \cap \mathcal{M}_{bia}$.

Proof. Let B be the triangular basis of Λ found by Algorithm [1](#page-10-0) with monomial ordering $\langle \mathbf{x} \rangle$ and $\mathbf{v} \in \Lambda$ satisfying the HHG bound. Let $g = \varphi^{-1}(\mathbf{x})$ be the corresponding polynomial. Assume g is not defined over $M \cap M_{\text{big}}$, and let m be the largest monomial in $M \setminus M_{\text{big}}$ with nonzero coefficient. The row in B corresponding to leading monomial m has Gram-Schmidt norm $||b_m^*|| \ge$ $m(\mathbf{X}) \ge P \ge p$, so because v is an integer linear combination of rows of B and the coefficient of m in g is nonzero, v includes at least one copy of b_m . Thus $||v|| \ge ||b_m^*||$, contradicting the claim that v satisfies the HHG bound.

In other words, it is unhelpful to consider any monomials not in \mathcal{M}_{big} ; this set is "big enough" to guarantee that the optimal dual lattice includes all sufficiently short vectors for a particular ideal (if any exist at all). Running Algorithm [1](#page-10-0) on \mathcal{M}_{big} therefore automatically produces optimal shift polynomials and a corresponding Coppersmith lattice, eliminating the need for hand-crafted strategies.

Depending on bounds X and P , the set \mathcal{M}_{big} , and therefore the lattice rank, may be large. While modern lattice reduction algorithms [\[39\]](#page-29-9) are often capable of reducing large lattices in practice, the large rank is computationally expensive and the lattice bounds may not guarantee that short vectors are found. To improve our choice of monomials further, we must examine the sublattice structure of optimal Coppersmith lattices.

5.2 Sublattice structure

Coppersmith lattices have a rich sublattice structure. This sublattice structure leads to one of the major open questions of prior research: Coppersmith's method often far outperforms expectations, finding lattice vectors significantly shorter than the determinant bound predicts. These unexpectedly short vectors belong to dense sublattices, or sublattices $\Lambda_{\text{sub}} \subset \Lambda$ where

$$
\det(\varLambda_{\mathrm{sub}})^{1/\mathrm{rank}(\varLambda_{\mathrm{sub}})} < \det(\varLambda)^{1/\mathrm{rank}(\varLambda)},
$$

so understanding the sublattice structure has two benets. 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.

There are limits to this approach. For example, some of the sublattice structure is due to the existence of a small modular root, and a direct construction of these sublattices would imply a more efficient solution to Coppersmith problems that does not involve lattice reduction. As described in Section [2.5,](#page-5-1) a small modular root directly implies the existence of a small vector in the primal lattice. When the root is small enough to be found by lattice reduction, it belongs to a dense sublattice of rank 1 in the primal lattice. By duality [\[35\]](#page-29-8), this implies the existence of a dense sublattice of rank $|S| - 1$ in the dual lattice. Although knowledge of this sublattice does not lead to a more efficient construction, it does explain the gap between theoretical predictions and practical performance in works such as [\[19\]](#page-28-1).

Additionally, the structure of the shift polynomial ideal can also explain some of the sublattice structure. If the optimal dual lattice for ideal J includes dense sublattices (such as due to the existence of a small modular root), then there is a polynomial $g \in J$ with unexpectedly small coefficients. The polynomial $g^2 \in J^2$ will also have small coefficients, implying the existence of dense sublattices in the optimal dual lattice for ideal J^2 . 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 works such as [\[12\]](#page-28-10), particularly the entries in Table 1 which report an LLL factor ≈ 0.5 .

While some of the sublattice structure is explained by the existence of a small modular root or exacerbated by the choice of shift polynomial ideal, it is not enough to fully explain the gap between theory and practice; much of the sublattice structure actually depends on the coefficient values of the shift polynomials. Fortunately, there is a straightforward way to investigate the remaining sublattice structure. First, replace the modulus p of a Coppersmith instance with a prime p' of the same size. This is to destroy any sublattice structure caused by small modular roots, as a root modulo p is no longer a root modulo p' . Second, build the optimal dual lattice for \mathcal{M}_{big} and reduce. Lattice reduction finds vectors that approximate the lattice minima, so the shortest vectors of the output basis span a dense sublattice. Although this is an a posteriori construction of a dense sublattice basis, we can examine this sublattice (by computing its Hermite Normal Form) for clues about performing a priori construction.

We have done this, and we find that there are two predominant explanations for sublattice structure due to the coefficient values of the shift polynomials.

5.3 Sparse polynomials and graph search

The use of normal forms relative to a Gröbner basis in line [9](#page-10-3) of Algorithm [1](#page-10-0) typically results in sparse shift polynomials. This means 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 $S = \{N, Nx, x^2 + a\}$ and $S_{\text{sub}} = \{N, x^2 + a\}.$

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

Basis B has determinant N^2X^3 and rank 3. Because of the sparsity of polynomials in S_{sub} , observe that the second column of B_{sub} is always zero (and thus 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 S contains hundreds or thousands of sparse polynomials. We must search over all possible subsets of 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 performs these steps, automatically and efficiently identifying these dense sublattices.

The requirement for all-zero columns is related to shift polynomial suitability. If S is (\mathcal{M}, \leq) -suitable, and $\mathcal{S}_{\text{sub}} \subset \mathcal{S}$ is $(\mathcal{M}_{\text{sub}}, \leq)$ -suitable, then this means \mathcal{S}_{sub} is defined over \mathcal{M}_{sub} , and the columns corresponding to monomials $\mathcal{M} \setminus$ \mathcal{M}_{sub} are necessarily all zero. Recall that a $(\mathcal{M},<)$ -suitable $\mathcal S$ has determinant $\det(A_{\mathcal{S}}) = \prod_{f \in \mathcal{S}} \text{LT}(f)(\mathbf{X})$ and rank \mathcal{S} . Our algorithm involves a directed graph that encodes information about shift polynomial suitability.

The directed graph represents dependencies between monomials in polynomials in S. Given a $(M, <)$ -suitable set of shift polynomials S, we define a graph G with vertices M and directed edges (m_1, m_2) if $m_1 \neq m_2$ and $\exists f \in S$ with $LM(f) = m_1$ and the coefficient of m_2 in f is nonzero. Once again, consider the $({1, x, x²}, <)$ -suitable $S = {N, Nx, x² + a}$. The corresponding graph follows.

Directed edges denote dependencies for $(\mathcal{M}_{sub}, <)$ -suitable subsets $\mathcal{S}_{sub} \subset \mathcal{S}$. If $x^2 \in M_{sub}$, then $x^2 + a \in S_{sub}$, implying $1 \in M_{sub}$. 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 $v_2 \in V \backslash V'$.

Picard [\[38\]](#page-29-12) 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 $(\mathcal{M}_{sub}, <)$ -suitable proper subset $\mathcal{S}_{sub} \subseteq \mathcal{S}$ with better determinant bounds if one exists. This process is documented in Al-gorithm [2.](#page-17-0) Intuitively, we want to find subsets with small determinant, or minimize the product $\prod_{f \in \mathcal{S}_{\text{sub}}} \text{LT}(f)(\bm{X})$. If we weight vertices by $-\log \text{LT}(f)(\bm{X}),$ then we search for closures which maximize the sum of weights. 3

Theorem [2](#page-17-0). Algorithm 2 is correct. On input X, M, S , it returns proper, nonempty subset \mathcal{M}_{sub} and corresponding $(\mathcal{M}_{sub}, <)$ -suitable $\mathcal{S}_{sub} \subsetneq \mathcal{S}$ where

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

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

Proof. Observe that any closure of G corresponds to a suitable subset of 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 $\mathcal{S}' \subset \mathcal{S}$,

$$
\begin{aligned} &\det(\varLambda_{\mathcal{S}'})^{1/|\mathcal{S}'|} < \det(\varLambda_{\mathcal{S}})^{1/|\mathcal{S}|} \Leftrightarrow \prod_{f \in \mathcal{S}'} \mathrm{LT}(f)(\boldsymbol{X}) < \prod_{f \in \mathcal{S}} \mathrm{LT}(f)(\boldsymbol{X})^{|\mathcal{S}'|/|\mathcal{S}|}\\ \Leftrightarrow \sum_{f \in \mathcal{S}'} \log_2 \mathrm{LT}(f)(\boldsymbol{X}) < \frac{|\mathcal{S}'|}{|\mathcal{S}|} \sum_{g \in \mathcal{S}} \log_2 (\mathrm{LT}(g)(\boldsymbol{X})) \Leftrightarrow 0 < \sum_{m \in \mathcal{M}'} w_m. \end{aligned}
$$

³ We actually seek to find small $\det(A_{S_{sub}})^{1/|S_{sub}|}$, which requires different weights.

Algorithm 2: SuitableSubset

Input : Bounds X, monomial set M and (M, \leq) -suitable S **Output:** $\mathcal{M}_{sub} \subseteq \mathcal{M}$ and $(\mathcal{M}_{sub}, \langle \rangle)$ -suitable $\mathcal{S}_{sub} \subseteq \mathcal{S}$ with better determinant bounds if one exists, else ⊥ 1 $E \leftarrow \{(m_1, m_2) \mid \exists f \in \mathcal{S} \text{ with } LM(f) = m_1 \text{ and } m_2 \neq m_1 \text{ is a monomial in } f\}$ 2 Construct directed graph $\mathcal{G} \leftarrow (\mathcal{M}, E)$ 3 for $f_m \in \mathcal{S}$ with $LM(f_m) = m$ do // Set weight of vertex $m \in \mathcal{M}$ $4 \mid w_m \leftarrow -\log_2(\text{LT}(f_m)(\boldsymbol{X})) + \frac{1}{|S|} \sum_{g \in S} \log_2(\text{LT}(g)(\boldsymbol{X}))$ 5 $\mathcal{M}_{sub} \leftarrow$ maximal closure of G with weights $\{w_m\}_{m \in \mathcal{M}}$ 6 if $\sum_{m \in {\cal M}_{sub}} w_m = 0$ then 7 return ⊥ 8 else 9 $\Big| \mathcal{S}_{\text{sub}} \leftarrow \{f \in \mathcal{S} \mid \text{LM}(f) \in \mathcal{M}_{\text{sub}}\}\$ 10 **return** M_{sub}, S_{sub}

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}_{sub} . If no such proper nonempty subset exists, then the maximal closure has total weight 0, and Algorithm [2](#page-17-0) returns ⊥.

To obtain $\mathcal{M}_{\text{heur}}$, we begin with \mathcal{M}_{big} . We find that inclusion of $f \in J_{\infty}$ in S leads to spurious short vectors that fail to satisfy Heuristic [1,](#page-6-1) so we set $\mathcal{M}' = \mathcal{M}_{\text{big}} \setminus \text{LM}(J_{\infty})$, which is easily computed from the Gröbner basis of J_{∞} . Next, we iteratively apply Algorithm [2](#page-17-0) to \mathcal{M}' until no more proper subsets are found. This final set is $\mathcal{M}_{\text{heur}}$.

5.4 Small coefficients in shift polynomials

In many cases, the graph-based process in Section [5.3](#page-15-0) works surprisingly well in practice, but there are some cases where it fails. In particular, shift polynomials with small, non-zero coefficients may lead to dense sublattices. Consider the lattice bases for $S = \{N, Nx, x^2 - x + a\}$ and $S_{sub} = \{N, x^2 - x + a\}$:

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

We can no longer eliminate a column of B_{sub} , so the determinant calculation is more involved, and the graph-based search fails to find this sublattice. In these situations, it is often helpful to perform unraveled linearization. For this example, we may introduce a new variable $u = x^2 - x$ with bound $U = X^2 + X$.

The lattice bases for $\mathcal{S}' = \{N, Nx, u + a\}$ and $\mathcal{S}'_{\text{sub}} = \{N, u + a\}$ are then

$$
B' = \begin{pmatrix} N & 0 & 0 \\ 0 & NX & 0 \\ a & 0 & U \end{pmatrix} \qquad B'_{\text{sub}} = \begin{pmatrix} N & 0 & 0 \\ a & 0 & U \end{pmatrix}
$$

and the graph-based search is expected to succeed.

6 Asymptotically fast use of precomputation

The monomial selection strategies in Section [5](#page-13-0) are powerful and effective, but the cost of Gröbner basis computation becomes increasingly expensive as the multiplicity grows. This can become problematic in practice due to longer running times, but it also makes it challenging to analyze the asymptotic behavior of Coppersmith's method. To remedy this, we propose a strategy based on symbolic precomputation of a set of shift polynomials. To extend this precomputed set to higher multipliicties, we replace the costly Gröbner basis computations with polynomial-time operations on polynomially sized sets. Our construction appears to satisfy a heuristic assumption by [\[33\]](#page-29-0), enabling us to use their polynomial interpolation strategy to heuristically determine the asymptotic behavior of our precomputation strategy. As a result, we obtain a shift polynomial strategy that is fast in practice while simultaneously enabling machine-generated proofs of asymptotic Coppersmith behavior.

6.1 Symbolic representation of shift polynomials

The algorithms of Section [4](#page-8-0) and Section [5](#page-13-0) can be performed symbolically when there is a single modulus p. We use variables 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 $\mathbb{Q}[c]$, which we denote by \mathbb{K}_c . We use variable N to represent the multiple of modulus p , and we specify input relations by the polynomial ring $\mathbb{K}_c[N, x]$. We build ideals, calculate Gröbner bases, and run Algorithm [1](#page-10-0) in this polynomial ring.

For example, consider the problem of factoring RSA modulus $N = pq$ when the least significant bits of p are known [\[34\]](#page-29-13). N is a known multiple of p, and we may use relation $f(N, x) = c_1x + c_2$ 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 $J = \langle N, f \rangle \subset \mathbb{K}_c[N, x]$ 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, Algorithm [1](#page-10-0) returns a symbolic representation of the shift polynomials S_1 . If J_{∞} is present, we also compute a symbolic representation of its Gröbner basis. During an attack, once the values of coefficients c are known, they may be substituted into the precomputed S . Division in fraction field \mathcal{K}_{c} is replaced by inversion modulo $N^{k_{\text{pre}}}$, so this substitution requires that the denominators in 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 S_1 which share a root modulo p and belong to an ideal $J \subset \mathbb{Z}[x]$. We use S_1 to compute shift polynomials with higher multiplicities to avoid calculating the Gröbner basis of $J^k + J_{\infty}$ 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 $\mathcal{S}_{k,\boldsymbol{t},\text{ul}}\subset J^k+J_\infty$. 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:

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

Computing polynomials in J^k . We recursively construct

$$
S_k = \Phi\left(\left\{ff'\mid (f,f')\in S_1\times S_{k-1}\right\}\right).
$$

This set grows similarly to the monomial sets from Jochemsz and May [\[22\]](#page-28-3) and Meers and Nowakowski [\[33\]](#page-29-0), which consider terms in f^k and $\prod_i f_i^k$ respectively.

Computing x-shifts in J^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

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

This is like the extended strategy in [\[22\]](#page-28-3) and the y-shifts of Boneh and Durfee [\[7\]](#page-27-0).

Unravelling into $J^k + J_\infty$. As described in Section [5.3,](#page-15-0) it is empirically helpful to exclude monomials in $LM(J_{\infty})$ when using unravelled linearization.

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

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

6.3 Specifying monomials for precomputation

In order to use Algorithm [1](#page-10-0) to find a symbolic representation of shift polynomials 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 [\[6\]](#page-27-7) to analyze bivariate integer Coppersmith problems, and recently Feng, Nitaj, and Pan [\[17\]](#page-28-11) used sumsets theory to connect optimal Coppersmith bounds to the volume of a Newton polytope.[4](#page-20-0)

Every monomial $\prod_{i=1}^\ell x_i^{e_i}$ maps to an integer point (e_1,\ldots,e_ℓ) in ℓ -dimensional space, and our definitions implicitly make use of this bijection. 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 Minkowski sum $\mathcal{P}_f + \mathcal{P}_g$ is the Newton polytope of product fg. We specify \mathcal{M}_1 using convex polytopes as well, and a set \mathcal{M}_{vert} of monomials representing the integer vertices of the polytope $\mathcal{P}_1 = \text{ConvexHull}(\mathcal{M}_{\text{vert}})$:

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

Since a multiple of the modulus is known, S_1 is $(\mathcal{M}_1, \langle \cdot \rangle)$ -suitable, and the leading monomials in S_1 correspond one-to-one with monomials in \mathcal{M}_1 , which correspond one-to-one with integer points in a polytope. The sets $\mathcal{S}_k, \mathcal{S}_{k,t}$, and $\mathcal{S}_{k,\mathbf{t},\mathrm{u}}$ also correspond to polytopes $\mathcal{P}_k,\mathcal{P}_{k,\mathbf{t}}$, and $\mathcal{P}_{k,\mathbf{t},\mathrm{u}}$.

- $-I$ F \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 Netwon 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 $\mathcal{P}_{k,t}$ is the union of translations of \mathcal{P}_k . The maximum translation in each dimension is t_i .
- $-LM(J_{\infty})$ corresponds to a union of cones, so $\mathcal{P}_{k,t,\text{ul}}$ is the (nonconvex) polytope of $\mathcal{P}_{k,t}$ minus the cones.

These polytopes are depicted in Figure [1.](#page-21-0)

6.4 Computing lattice properties with polytopes

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

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

⁴ An October, 2024 revision of their work introduces a polytope-based shift polynomial strategy, and introduces the same foundational results of Ehrhart, Brion and Vergne as in the proof of our Lemma [6.](#page-21-1) While this section references the prior revision of Feng et al.'s work, we call attention to this new overlap with results in concurrent work. However, we note that Feng et al.'s new strategy uses input polynomials to define a convex polytope, then selects shift polynomials based on dilations of the polytope. Our strategy is more general and uses a convex polytope to specify shift polynomials, then directly constructs shift polynomials of higher multiplicity.

Fig. 1: Polytopes corresponding to our shift polynomial sets. Polytope \mathcal{P}_1 is defined by vertices $\mathcal{M}_{\text{vert}} = \{1, x_1^3, x_1^3 x_2, x_1 x_2^2, x_2^2\}$, and $J_{\infty} = \langle x_1^2 x_2 - x_1 \rangle$. These correspond to the monomials that appear in S_1 , S_2 , $S_{2,(0,1)}$, and $S_{2,(0,1),ul}$. Polytope P_2 is P_1 scaled by 2. Polytope $P_{2,(0,1)}$ is P_2 along with monomials from x_2 shifts. $\mathcal{P}_{2,(0,1),\text{ul}}$ eliminates all monomials that are multiples of $x_1^2x_2$.

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

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

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

$$
\dim \Lambda_{\mathcal{S}_{k,\mathbf{t},\mathbf{u}1}} = s_{\dim}(k,\mathbf{t})
$$

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

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 its *Ehrhart polynomial*. As a result, we may use Ehrhart theory to bound the complexity of our precomputation strategy. We refer to the introduction of [\[10\]](#page-28-12) for background on Ehrhart polynomials and the theorems we cite in this proof.

Lemma 6. Let \mathcal{P}_1 and \mathcal{S}_1 be a convex polytope and shift polynomial set as previously defined. Then $|S_k|$ is polynomial in k. Additionally, if $J_{\infty} = \{0\}$ and we fix $\boldsymbol{t} = \boldsymbol{0}$, then $s_{dim}(k, \boldsymbol{0})$ and $s_{x_i}(k, \boldsymbol{0})$ 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 de-scribed by the Ehrhart polynomial [\[15\]](#page-28-13) 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 $\mathcal{S}_{k,t,\text{ul}} = \mathcal{S}_{k,t} = \mathcal{S}_k$, $s_{\text{dim}}(k,0) = |\mathcal{S}_k|$ is polynomial in k.

We have $s_{x_i}(k, \mathbf{0}) = \sum_{\mathbf{e} \in k\mathcal{P}} e_i$, which is a sum over integer points in a dilated polytope, weighted by a homogenous polynomial of degree 1. A result of Brion and Vergne [\[9\]](#page-28-14) 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 $|\mathcal{S}_{k,t}| \leq (\max t)^{\ell} |\mathcal{S}_k|$ is bounded by a polynomial in k and t, and $|\mathcal{S}_{k,t,\text{ul}}|$ is also asymptotically polynomial. This means that $S_{k,t,\text{ul}}$ 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 [\[33\]](#page-29-0), is well supported by experiment, and was recently partially justified in theory by Feng, Nitaj, and Pan [\[17\]](#page-28-11).

Heuristic 2 Let S_1 represent a convex polytope, and let $S_{k,t,ul}$ be as previously defined. Then if $\mathcal{S}_{k,t,ul}$ 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 t and have maximum total degree $\ell + 1$.

As an example of these polynomials, consider the polytope in Figure [1.](#page-21-0) 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

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

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

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.

Lemma 7. For a particular multivariate Coppersmith problem, assume each bound X_i is parametrized 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 S_1 with shared root modulo p, let s_{dim} , s_{x_i} , and s_{C_j} represent the functions describing the parametrized lattice dimension and determinant. Assume Heuristics [1](#page-6-1) and [2](#page-22-0) hold, and assume the leading coefficients in S are proportional to p. For sufficiently large p and any $\tau \geq 0$, the Coppersmith problem is solvable for $\delta < \delta^* - \epsilon$ where $\delta^* =$

$$
\lim_{k \to \infty} \frac{ks_{dim}(k, k\tau) - \sum_{i=1}^{\ell} a_i s_{x_i}(k, k\tau) - \sum_{C_j \in \text{LC}(S_1)} s_{C_j}(k, k\tau) \log_p C_j}{\sum_{i=1}^{\ell} b_i s_{x_i}(k, k\tau)} \tag{3}
$$

In particular, for multiplicity k we require

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

.

Excluding the final root recovery step, this takes time polynomial in ϵ^{-1} .

Proof. Equation [3](#page-22-1) sets $t = k\tau$ and combines Heuristic [2](#page-22-0) with heuristic inequal-ity [1.](#page-6-2) Since the limit converges polynomially quickly, we may take $k = \Theta(\epsilon^{-1})$ and $\log p = \Theta(\text{poly}(k))$. By Lemma [6,](#page-21-1) $S_{k,t,\text{ul}}$ 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,](#page-23-1) which symbolically determines asymptotic bounds for a given multivariate Coppersmith problem. We use Section [6.1](#page-18-1) to compute a symbolic ideal, Section [6.3](#page-19-0) to find \mathcal{M}_1 , Algorithm [1](#page-10-0) to find S_1 , Section [6.2](#page-19-1) to find symbolic lattice constructions, interpolation to determine s_{dim} and the other polynomials, and Lemma [7](#page-22-2) to find asymptotic bounds.

This algorithm, which adapts our methods to the approach of Meers and Nowakowski [\[33\]](#page-29-0), allows one to compute asymptotic Coppersmith bounds in a fully automated way. While this approach still requires careful choice of $\mathcal{M}_{\text{vert}}$, we believe that it greatly simplifies the process of proving and verifying asymptotic bounds for multivariate Coppersmith problems.

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.](#page-24-0) Since there are tradeoffs between recoverable boundary, lattice dimension, and running time, it is challenging to directly compare compare two shift polynomial selection strategies. We therefore focus on the following attributes to demonstrate the capabilities of our new methods.

- **Bounds.** For any particular Coppersmith problem, we want to maximize the size of the recoverable bounds \boldsymbol{X} . For a particular multiplicity, we experimentally determine the maximum \boldsymbol{X} solved by the shift polynomial strategy in prior work and the maximum solved by our optimal shift polynomial strat-egy in Section [4](#page-8-0) and proven monomial strategy \mathcal{M}_{biq} in Section [5.1.](#page-13-1)
- **Dimension.** A smaller lattice dimension typically leads to faster lattice reduction. For a particular multiplicity, we calculate the rank of the lattice in the prior work's shift polynomial strategy. We run the graph-based shift polynomial strategy in Section [5](#page-13-0) and record the rank of our lattice.
- $-$ Time. For practical attacks, running time can be important. For a particular multiplicity, we compare their concrete running time against our concrete running time using our asymptotically fast precomputation-based strategy in Section [6.](#page-18-0)
- **Asymptotics.** A regular feature of Coppersmith papers is asymptotic analysis to determine the maximum recoverable bound 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.](#page-23-2)

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 $(\vee\vee)$, equivalent to (\vee) , or worse (\times) 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.](#page-31-0)

Coppersmith Problem	Prior	Bounds	Dimension	Time	Asymp.
CIHNP-CSIDH	[33]	\vee same	\vee same	\vee same	\vee same
CIHNP-CSURF	$\left[33\right]$	\vee better	X worse	$\mathbf{\triangledown}$ same	\vee better
MIHNP	$\left[48\right]$	\vee better	X worse	$\mathbf{\triangledown}$ same	\vee better
ECHNP	[49]	\vee better	X worse	X worse	\vee better
Stereotyped RSA	[28]	\vee same	\vee same	$\mathbf{\triangledown}$ same	$\mathbf{\triangledown}$ same
Partial factoring RSA	$\left[28\right]$	\vee same	\vee same	\vee same	\vee same
Partial ACD	$[12]$	\vee same	\vee same	\vee same	\vee same
Small RSA priv. exp. d	$\left[20\right]$	\vee same	\vee better	\vee same	\vee same
RSA Power Gen.	$\left[19\right]$	\vee same	\vee better	\vee better	$\mathbf{\triangledown}$ same
Small CRT RSA d_n, d_q	[43]	\vee same	\vee better	X worse	X worse
Partial CRT-RSA d_p, d_q	[30]	\vee same	$\mathbf{\triangledown}$ same		
SMUPE	[32]	\vee better	\vee better		
Common prime RSA	$\left[25\right]$	\vee same	\vee same	$\mathbf{\triangledown}$ same	\vee same
Small Multipower RSA d	$\left[25\right]$	\vee same	\vee same	$\mathbf{\triangledown}$ same	\vee same

A number of large-scale trends are apparent in Table [1.](#page-24-0) First, our provable strategy always performs as well as or better than prior strategies when it comes to maximizing recoverable bounds. This demonstrates the reliability of our optimal shift polynomial selection strategy of Algorithm [1](#page-10-0) and our provably good monomial selection in Section [5.1.](#page-13-1) Second, our graph-based shift polyno-mial strategy of Algorithm [2](#page-17-0) frequently finds smaller, and therefore more easily reduced, lattices. In the cases where this algorithm 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 $\mathcal{M}_{\mathrm{vert}}$, 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 as described in Section [6.6](#page-23-2) often leads to the same bounds as in prior work. However, our Algorithm [3](#page-23-1) is also sensitive to choice of $\mathcal{M}_{\text{vert}}$, and does not always lead to the same bound.^{[5](#page-25-0)}

Full numerical data and additional details about the diverse array of problems we examined are availible in Appendix [A.](#page-31-0) 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 a number of previously studied multivaraiate Coppersmith problems. In particular, we evaluated the maximum bounds X which are solvable at least 50% of the time for particular choices of a 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 strategy that uses all monomials in \mathcal{M}_{big} from Lemma [5,](#page-13-2) the graph-based strategy in Section [5.3,](#page-15-0) and the precomputation strategy in Section [6.2.](#page-19-1) 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](#page-25-1)} for Gröbner basis computations in $\mathbb{Q}[x]$, and flatter^{[7](#page-25-2)} 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 [\[8\]](#page-27-2). Xu et al. revisited the problem in [\[48\]](#page-30-3).

⁵ An October, 2024 revision of Feng, Nitaj, and Pan's work [\[17\]](#page-28-11) improves the existing asymptotic bounds for CSIDH-CSURF beyond both Meers and Nowakowski's work and our own. This highlights the sensitivity of our approach, as the bounds achieved for a particular choice of $\mathcal{M}_{\text{vert}}$ are not guaranteed to be optimal.

 6 <https://msolve.lip6.fr/>

 $\frac{7 \text{ https://github.com/keeganryan/flatter}}{2}$

For MIHNP, the ℓ input relations have the form

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

and share a root modulo a known prime p.

We report the results for MIHNP with four relations in Table [2.](#page-26-0) The shift strategy in [\[48\]](#page-30-3) 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	
κ								$\lg X_i$ Dim. Time $\lg X_i$ Dim. Time $\lg X_i$ Dim. Time $\lg X_i$ Dim. Time					
		332		4.0 II	373	16	0.5	373	11	(0.4)	\blacksquare		
$\overline{2}$		405	16	4.4	410	86	12.2	410	50	6.9	405	16	1.5
-3		\overline{a}		\blacksquare	446	296	211.8	446	76	66.0	\blacksquare		
4		$\overline{}$				\blacksquare		460	221	581.3	442	81	25.71
	Existing bound: $\log_p X_i = 0.2432$ Our bound: $\log_p X_i = 0.5208$												

We ran Algorithm [3](#page-23-1) 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 weighted monomial ordering recognizes $\alpha \gg x_i$, and Algorithm [1](#page-10-0) automatically finds shift polynomials with small leading monomial, or shift polynomials that don't include α.

This example also illustrates a drawback of our approach. While the approach in [\[48\]](#page-30-3) provides asymptotic analysis for $\ell \to \infty$, our approach only applies to fixed values of ℓ , and as ℓ increases, Gröbner basis calculations become probitively expensive.

We also observe that the strategy of using all monomials in \mathcal{M}_{big} 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 solving multivariate polynomials with known modulus, our graph-based methods are ineffective against integer Coppersmith problems or the General Approximate Common Divisors problem of Cohn and Heninger [\[12\]](#page-28-10) where no multiple of the modulus is known; the challenge is calculating the determinant of Coppersmith lattices that are not full-rank. Our approach also does not capture the multi-step approaches used by Peng et al. [\[37\]](#page-29-14) or May et al. [\[30\]](#page-29-4), 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 for providing feedback on a draft 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:](https://doi.org/10.1007/978-3-642-31448-3_28) [//doi.org/10.1007/978-3-642-31448-3_28](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) https://doi.org/10.1007/978-1-4612-0913-3_11
- 4. Bleichenbacher, D., May, A.: New attacks on RSA with small secret CRTexponents. 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.](https://doi.org/10.1007/11745853_1) [org/10.1007/11745853_1](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_](https://doi.org/10.1007/11426639_15) [15](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/](http://www.jstor.org/stable/2152855) [2152855](http://www.jstor.org/stable/2152855)
- 10. Chen, B.: Ehrhart polynomials of lattice polyhedral functions (2005). [https://doi.](https://doi.org/10.1090/conm/374/06898) [org/10.1090/conm/374/06898,](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/](https://doi.org/10.1007/978-3-662-53887-6_28) [10.1007/978-3-662-53887-6_28](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/](https://doi.org/10.1007/11426639_22) [10.1007/11426639_22](https://doi.org/10.1007/11426639_22)
- 17. Feng, Y., Nitaj, A., Pan, Y.: New results for coppersmith's method from the perspective of sumsets 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/](https://doi.org/10.1007/978-3-642-10366-7_29) [10.1007/978-3-642-10366-7_29](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 nding 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 CRTexponents 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/](https://doi.org/10.1007/978-3-540-74143-5_22) [10.1007/978-3-540-74143-5_22](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/](https://doi.org/10.1007/BF01457454) [10.1007/BF01457454](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. In: Nguyen and Vallée [\[36\]](#page-29-15), pp. 315348. <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). $\frac{htps}{/}{\text{doi.org/10}}$. [1007/978-3-030-92062-3_4](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.) ASI-ACRYPT 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. In: Nguyen and Vallée [\[36\]](#page-29-15), pp. 1969. <https://doi.org/10.1007/978-3-642-02295-1>
- 36. Nguyen, P.Q., Vallée, B. (eds.): The LLL Algorithm - Survey and Applications. ISC, Springer (2010). <https://doi.org/10.1007/978-3-642-02295-1>
- 37. 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
- 38. 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/](https://doi.org/10.1287/mnsc.22.11.1268) [mnsc.22.11.1268,](https://doi.org/10.1287/mnsc.22.11.1268) <https://doi.org/10.1287/mnsc.22.11.1268>
- 39. 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/](https://doi.org/10.1007/978-3-031-38548-3_1) [978-3-031-38548-3_1](https://doi.org/10.1007/978-3-031-38548-3_1)
- 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. 473484. Springer, Berlin, Heidelberg (Jun 2009). [https://doi.org/10.1007/](https://doi.org/10.1007/978-3-642-01957-9_29) [978-3-642-01957-9_29](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/](https://doi.org/10.1007/978-3-319-13051-4_21) [978-3-319-13051-4_21](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/](https://doi.org/10.1007/s10623-019-00685-y) [s10623-019-00685-y](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:](https://doi.org/10.1007/s10623-017-0435-4) [//doi.org/10.1007/s10623-017-0435-4](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), 53375356 (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.) ASI-ACRYPT 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 [\[33\]](#page-29-0). 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

$$
f = (c_1 + x_1)(c_2 + x_2) + 2(c_1 + x_1) - 2(c_2 + x_2) + 12
$$

\n
$$
g = (c_3 + x_3)(c_2 + x_2) + 2(c_2 + x_2) - 2(c_3 + x_3) + 12
$$

\n
$$
h = (c_1 + x_1)(c_3 + x_3) - 2(c_1 + x_1) + 2(c_3 + x_3) + 12
$$

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

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

\n
$$
g = (c_3 + x_3)^2 + 12(c_3 + x_3) - 4(c_3 + x_3)(c_1 + x_1)^2 - 8(c_1 + x_1)^2 + 36.
$$

We ran our shift polynomial strategies on CI-HNP for both CSIDH and CSURF, and report the results in Table [3](#page-31-1) and Table [4](#page-32-0) respectively.

Table 3: CI-HNP for CSIDH. Largest solvable bound for 512-bit modulus p. Note that [\[33\]](#page-29-0) is only compatible if the multiplicity is divisible by 3.

We ran Algorithm [3](#page-23-1) 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

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

This matches the existing bounds of $X_i < p^{11/24}$ [\[33\]](#page-29-0).

Table 4: CI-HNP for CSURF. Largest solvable bound for 512-bit modulus p. Note that [\[33\]](#page-29-0) is only compatible if the multiplicity is divisible by 2. In the all-monomial strategy with $k = 2$, we have $X^{11} \approx p^2$, so whichever value is larger determines $|\mathcal{M}_{\text{big}}|$. We report the maximum size, which is 364.

		33			All monoms.			Graph search			Precomputed	
\boldsymbol{k}			$\lg X_i$ Dim. Time $\lg X_i$ Dim. Time $\lg X_i$ Dim. Time $\lg X_i$ Dim. Time									
		\blacksquare		71	120	8.6II	69	48	7.01	$\overline{}$		
$\overline{2}$	74	33	1.7	93	364	258.3	91	145	119.5	74	34	2.3
3				103	680	2554.7	101	324	1124.8	\blacksquare		
$\overline{4}$	96	165	111.0			-			$\qquad \qquad \blacksquare$	97	173	110.7
						Existing bound: $\log_p X_i = 0.2439$ Our bound: $\log_p X_i = 0.2528$						

We ran Algorithm [3](#page-23-1) on CIHNP-CSURF with input

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

$$
\mathcal{M}_{\text{vert}} = \{1, x_1^4 x_3, x_3^3, x_1^4, x_1^3 x_2^2 x_3, x_1^3 x_2^2, x_2^2, x_1^2 x_3^2, x_2^2 x_3^2, x_1 x_2^2 x_3^3\}
$$

and got output

$$
\boldsymbol{\tau} = (0, 0, 0), \quad \delta < 0.2528.
$$

This improves Meers and Nowakowski's bounds [\[33\]](#page-29-0) of $X_i < p^{10/41.8}$ $X_i < p^{10/41.8}$ $X_i < p^{10/41.8}$

A.2 Modular Inversion Hidden Number Problem

Boneh, Halevi, and Howgrave-Graham introduced the Modular Inversion Hidden Number Problem (MIHNP) in 2001 [\[8\]](#page-27-2). Xu et al. revisited the problem in [\[48\]](#page-30-3). For MIHNP, the ℓ input relations have the form

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

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,](#page-33-0) Table [6,](#page-33-1) and Table [7](#page-33-2) respectively. The shift strategy in [\[48\]](#page-30-3) is parameterized by (n', d', t') , which using our variable names is $(\ell-1, k, t)$. Their strategy is only defined for $0 \le t \le k$ and $k \le$ $\ell-2$, 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,](#page-30-3) Section VI.A].

⁸ An October, 2024 revision of Feng, Nitaj, and Pan's work [\[17\]](#page-28-11) improves the bounds for CIHNP-CSURF to $\log_p X_i < 8/31 \approx 0.2580$.

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

			[48]							All monoms. Graph search Precomputed			
										\overline{k} t g X_i Dim. Time lg X_i Dim. Time lg X_i Dim. Time lg X_i Dim. Time			
		248	4				4.6 331 21 0.6 332 8			0.4	\sim		
$\overline{2}$	$\overline{1}$ $\overline{2}$	$\epsilon = \frac{1}{2} \left(1 - 1 \right)$			371		67 5.6 372			26 3.7 372		-27	4.4
3	$\overline{}$	L.			- 390	187				85.6 390 60 27.8			
4					$- 401 $					386 837.9 401 115 152.1 401		125	- 74.4
										Existing bound: $\log_n X_i = 0.0000$ Our bound: $\log_n X_i = 0.4444$			

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

			[48]			All monoms.				\parallel Graph search \parallel Precomputed		
k_{\parallel}									$ t g X_i $ Dim. Time $ g X_i $ Dim. Time $ g X_i $ Dim. Time $ g X_i $ Dim. Time			
	llol	332	7	4.0 373		16 0.5 373		-11	0.4			
$\overline{2}$		405	$16-1$	$4.4 $ 410	86		12.2 410	50 ₁		6.9 405	-16	1.5
3				446		296 211.8 446		-76 -	66.0II			
4						\blacksquare	460	221	$ 581.3 $ 442		- 81	25.7
									Existing bound: $\log_n X_i = 0.2432$ Our bound: $\log_n X_i = 0.5208$			

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

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

We ran Algorithm [3](#page-23-1) on MIHNP with $\ell \in \{3, 4, 5\}$ samples with input

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

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

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 [\[41\]](#page-30-9) and studied by Xu et al. in 2022 [\[49\]](#page-30-5). 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

$$
A_i = h_i(h_0 - x_{Q_i})^2 - 2h_0^2 x_{Q_i} - 2(a + x_{Q_i}^2)h_0 - 2ax_{Q_i} - 4b
$$

\n
$$
B_i = 2(h_i(h_0 - x_{Q_i}) - 2h_0x_{Q_i} - a - x_{Q_i}^2)
$$

\n
$$
C_i = h_i - 2x_{Q_i}
$$

\n
$$
D_i = (h_0 - x_{Q_i})^2
$$

\n
$$
E_i = 2(h_0 - x_{Q_i}),
$$

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,](#page-34-0) [9,](#page-34-1) and [10](#page-35-0) respectively. The shift strategy in [\[49\]](#page-30-5) 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,](#page-30-5) Section 4.2].

Table 8: ECHNP with 3 samples. We consider 256-bit modulus p . The shift strategy in [\[49\]](#page-30-5) is parameterized by (n', d', t') , which using our variable names is $(3, k, t)$.

			49			All monoms.				Graph search	Precomputed		
\boldsymbol{k}										$t \lg X_i $ Dim. Time $ \lg X_i $ Dim. Time $ \lg X_i $ Dim. Time $ \lg X_i $ Dim. Time			
		60	14	0.6 ₁	64	35		64	19	1.31	58		5.9
$\overline{2}$	ച	68	32	2.4			102.21	78		58.1	64	108	33.0
-3		68	48	3.3 ^{II}	85	715	2418.1	83	397	1217.1	66	366	569.8
	Existing hound \log $Y = 0.2083$ $Q_{\rm HF}$ bound \log												

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 [\[49\]](#page-30-5) 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		
κ							$\lg X_i$ Dim. Time $\lg X_i$ Dim. Time $\lg X_i$ Dim. Time				
		69	22	1.31		56 -	3.51	72			
		79	60	9.8	87	252	215.1	87		130.3	
	${\mathbf E}$ visting hound \cdot log										

Existing bound: $\log_p X_i = 0.2772$

Table 10: $\bf ECHNP$ with 5 samples. We consider 256-bit modulus p. The shift strategy in [\[49\]](#page-30-5) 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			
\boldsymbol{k}							$ 2 \lg X_i $ Dim. Time $ \lg X_i $ Dim. Time $ \lg X_i $ Dim. Time					
	2.3 77 32 84 78 5.9 75											
	462 $ 1301.5 $ 21.9 92 309 895.2 87 92 94											
	Existing bound: $\log_p X_i = 0.3224$											

We ran Algorithm [3](#page-23-1) on MIHNP with 3 samples with input

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

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

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 [\[13\]](#page-28-0). 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 [\[28\]](#page-29-2) 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 [\[28\]](#page-29-2), 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.](#page-35-1)

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

	[28]				All monoms.			Graph search			Precomputed			
\boldsymbol{k}		$\lg X$	Dim.	Time	$\lg X$	Dim.	Time	$\lg X$		$ Dim.$ Time	$\lg X$	Dim.	Time	
	$\overline{2}$	199	5	0.01	199	6	0.1	199	6	0.2	166	4	0.1	
$\overline{2}$	$^{\circ}2$	249	8	0.1	249	9	0.3 _l	249	9	0.2	237	7	0.1	
4	$\overline{2}$	285	14	0.4 _l	285	15	0.7	285	15	0.8	281	13	0.4	
6	$\overline{2}$	299	20	0.9	299	21	$1.6\,$	299	21	1.6	297	19	0.9	
8	$\overline{\mathbf{2}}$	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	$3\,21$	57	33.9	321	57	34.6	320	55	14.3	
20	$^{^{\prime}2}$	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](#page-23-1) on the stereotyped RSA problem with input

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

and got output

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

This agrees with existing asymptotic bounds of $X < N^{1/3}$ [\[13\]](#page-28-0).

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 has the correct 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

$$
N \equiv 0 \pmod{p}
$$

$$
x + a \equiv 0 \pmod{p}.
$$

We use the shift polynomial strategy by May [\[28\]](#page-29-2), 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.](#page-36-0)

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

	28				All monoms.			Graph search			Precomputed		
\boldsymbol{k}				$\lg X$ Dim. Time $\lg X$ Dim. Time $\lg X$ Dim. Time $\lg X$ Dim. Time									
	2	340	3	0.01	340	4	0.11	340	4	0.01	340	3	0.1
$\frac{5}{2}$	6	464	11	0.3	464	12 ¹²	0.5	464	12 ¹²	0.51	464	11	0.4
10 II		486	21	1.911	486	22	3.1	486	22	2.9	486	21	2.1
15 16		494	31	7.7	494	32	12.3	494	32	11.51	494	31	9.1
20 21		498	41	18.41	498	42	29.0ll	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: $\log_n X = 0.5$ Our bound: $\log_n X = 0.5$

We ran Algorithm [3](#page-23-1) 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 $X < N^{1/4}$ [\[28\]](#page-29-2).

A.6 Partial Approximate Common Divisors

Heninger and Cohn studied the Partial Approximate Common Divisors (PACD) problem in [\[12\]](#page-28-10). 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,](#page-37-0) [14,](#page-37-1) and [15](#page-37-2) respectively. In all cases, we used 1000-bit N and 400-bit p .

	12 $\lceil \lg X_i \rceil$ Dim. Time $\lVert \lg X_i \rVert$ Dim. Time $\lVert \lg X_i \rVert$ Dim. Time $\lVert \lg X_i \rVert$ Dim. Time				All monoms.			\parallel Graph search \parallel Precomputed						
\boldsymbol{k}														
	3	99	4	0.01	99	$5 -$	0.3 _l	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	4.8II	150	26	4.0	
		$15 37 $ 153	38	11.9	153	40		15.3 153	40	15.2	153	39	12.6	
		$\left\vert 20\right\vert \left\vert 50\right\vert$ 155 $\left\vert$	51	27.6	155	52	42.3	155	52	42.31	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 13: **PACD** with $\ell = 1$ sample.

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

			[12]		All monoms.			\parallel Graph search \parallel Precomputed					
k				t $\lg X_i$ Dim. Time $\lg X_i$ Dim. Time $\lg X_i$ Dim. Time $\lg X_i$ Dim. Time									
	$\vert 2 \vert$	174	6		0.1 174		$6 \mid 0.3 \mid 174 \mid$		6	0.2	\blacksquare		
3	5°	-216		$21 \mid 1.1 \mid$	$216\,$		$21 \mid 1.4 \mid 216 \mid$			$21 \mid 1.4 \mid 216$		26 l	1.5
-6		$ 10 $ 231	66		$20.8 $ 231	66 -		$23.1 $ 231	66	23.0	-231	64	20.2
9 ¹		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.

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

We ran Algorithm [3](#page-23-1) on PACD with $\ell = 1$ sample with input

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

and got output

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

We also ran Algorithm [3](#page-23-1) on PACD with $\ell \in \{2,3\}$ samples with input

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

and got output

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

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 [\[7\]](#page-27-0). 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 [\[20\]](#page-28-5).

We ran our shift polynomial strategies on the RSA with small private exponent problem and report the results in Table [16.](#page-38-0)

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 $|\mathcal{M}_{\text{big}}|$. 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_{\parallel}					$ t \lg X_1 $ Dim. Time $\lg X_1 $ Dim.		$\text{Time} \vert \vert \vert g X_1 \vert \text{Dim.} \vert \text{Time} \vert \vert \vert g X_1 \vert \text{Dim.} \vert \text{Time} \vert$						
$\overline{2}$		243	7	0.3 ₁	240	40	1.5						
3		259	11	0.3 ₁	259	83	7.7	259	8	0.91	259	15	0.5
4	12	263	19	1.1	263	174	66.1	263	14	2.3II	\blacksquare		
5	12	267	27	1.8	267	308	432.0	267	22	3.8II	\blacksquare		
6	3	270	37	3.8 ^l	270	493	2391.7	270	31	7.2	270	40	3.5
	13	272	48	7.5	271	746	5418.1	272	42	13.3	\blacksquare		
8	Ι3	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](#page-23-1) on the small RSA private exponent problem with input

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

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

and got output

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

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 [\[7\]](#page-27-0). 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 [\[19\]](#page-28-1). In this problem, an adversary is given the mostsignificant 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

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

Herrmann and May introduce the concept of unravelled linearization, adding

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

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](#page-40-0) and Table [18](#page-40-1) 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 $\ \lg X_i \ \text{Dim.} \text{Time} \text{lg }X_i \ \text{Dim.} \text{Time} \text{lg }X_i \ \text{Dim.} \text{Time} \text{lg }X_i \ \text{Dim.} \text{Time}$							
$\mathbf{1}$	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	1.0	378	95	16.5	378	10	1.0 ll	\sim			
4	385	25	0.8	385	161	97.1	385	15	2.0 _{II}	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](#page-23-1) 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 [\[19\]](#page-28-1).

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

		$\left\lceil 19\right\rceil$			All monoms.			Graph search		Precomputed		
\boldsymbol{k}		$\lg X_i$ Dim.	Time lg X_i Dim. Time lg X_i Dim. Time lg X_i Dim. Time									
			\overline{a}	339	28	1.0	340	6	0.41			
$\overline{2}$	392	7	0.3	392	108	15.5	392	$\overline{7}$	1.3 _l			
3	397	22	1.7	402	308	459.7	402	21	5.0 ₁			
4	413	39	1.71			\blacksquare	413	22	16.3	413	22	2.2
5	419	62	10.5			$\qquad \qquad \blacksquare$	423	50	77.9			
6	427	93	61.2			۰	429	93	172.21			
7	430	132	448.5			$\overline{}$	433	95	369.7			
8	433	181	5910.4			$\qquad \qquad \blacksquare$	437	159	862.7	435	95	50.71
	Existing bound: $\log_N X_i = 0.4615$ Our bound: $\log_N X_i = 0.4615$											

We ran Algorithm [3](#page-23-1) 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 [\[19\]](#page-28-1).

A.9 RSA-CRT with small private exponents

We consider the problem of RSA-CRT with small private exponents, first explored in [\[26\]](#page-29-3), with the best current results due to Takayasu, Lu, and Peng [\[43\]](#page-30-6). 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 $ed_q \equiv 1$ (mod $q-1$). This is rewritten as the following relations

$$
-1 - x_3(x_1 - 1) \equiv 0 \pmod{e}
$$

$$
-1 - x_4(x_2 - 1) \equiv 0 \pmod{e}
$$

$$
x_1x_2 - N = 0.
$$

with shared root $(p, q, \frac{ed_p-1}{p-1})$ $\frac{d_p-1}{p-1}, \frac{ed_q-1}{q-1}$ $\frac{a_q-1}{q-1}$) 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 unraveled linearization

$$
u_1 = x_1 + x_2
$$

\n
$$
u_2 = x_3x_1 + x_4x_2 + 2
$$

\n
$$
u_3 = x_3 + x_4 - 1
$$

\n
$$
u_4 = x_3x_4
$$

\n
$$
u_5 = x_3x_4x_1 + x_3x_4x_2 - x_3x_1 - x_4x_2 + x_3 + x_4 - 1
$$

and ran our shift polynomial strategies on the problem of RSA with small CRT exponents and report the results in Table [19.](#page-41-0)

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 [\[43\]](#page-30-6) yields a lattice of dimension 177 or 179 for $k = 8$, so we report the smaller value.

		43			Graph search		Precomputed			
\boldsymbol{k}			$Dim.$ Time	$\text{log}\frac{X_3}{X_1}\overline{\left \text{Dim.}\right }$		Time	lg		Dim. Time	
$\overline{4}$	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	
	$\mathbf{1}$ and $\mathbf{1}$		X_{2}	0.1000		\cap 1 1			Xэ	

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](#page-23-1) on the small RSA-CRT exponent problem with input

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

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

$$
\mathcal{M}_{\text{vert}} = \{1, u_1, u_2^2, u_3^2, u_4^2, u_5^2, u_1 u_3^2, u_1 u_5^2, u_2 u_3, u_2 u_4, u_2 u_5\}
$$

We fixed $\tau = 0$ because Algorithm [3](#page-23-1) was too slow otherwise and got output

 $\delta < 0.0486$.

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

As seen in Table [19,](#page-41-0) even though \mathcal{M}_{big} was too large to run our strategy on all monomials, our graph-based search algorithm found sublattices of approximately half the dimension of [\[43\]](#page-30-6). 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 [\[30\]](#page-29-4). 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

$$
x + (ec_1 + a - 1)(2^{-b}e^{-1} \mod aN) \equiv 0 \pmod{ap}
$$

$$
a \equiv 0 \pmod{a}
$$

$$
N \equiv 0 \pmod{p}.
$$

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.](#page-42-0)

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 [\[30\]](#page-29-4). Since the input relations have mixed moduli, we don't run our precomputation strategy.

		[30]			All monoms.			Graph search	
\boldsymbol{k}	$\lg X_1$	Dim.	Time	$\lg X_1$		$ \text{Dim.} \text{Time} \text{lg }X_1 $		Dim.	Time
1, 1	60	$\overline{2}$	0.0	210	3	0.1	210	3	0.3
2, 1	231	3	0.0	231	3	0.21	231	3	0.4
4, 2	265	$\bf 5$	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	28.21	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	$\bf{304}$	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) [\[32\]](#page-29-11). 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

$$
f_1 = (x + a_1)^3 - c_1 \equiv 0 \pmod{N_1}
$$

\n
$$
f_2 = (x + a_2)^5 - c_2 \equiv 0 \pmod{N_2}.
$$

We ran our shift polynomial strategies on SMUPE and report the results in Table [21.](#page-43-0) 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			
\boldsymbol{k}	$\lg X$		$Dim.$ Time	$\lg X$		Dim. Time	$\lg X$		Dim. Time	
			\blacksquare	292	8	0.4	292	8	0.2	
			\blacksquare	369	9	0.3 _l	369	9	0.3	
$\mathbf{2},$ $\left(2\right)$			-	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)			$\overline{}$	443	17	2.2 ₁	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 [\[32\]](#page-29-11). This is because our method finds novel shift polynomials, such as $\{f_1f_2, N_1f_2, N_2f_1, N_1N_2\}$, which all share a root modulo N_1N_2 .

A.12 Common Prime RSA

In 2014, Lu et al. studied the Common Prime RSA problem [\[25\]](#page-29-1). In this problem, the factors p, q of RSA modulus N have multiplicative orders which share a common prime g. That is, $p = 2aq + 1$ and $q = 2bg + 1$. The public and private exponents are e and d respectively. Lu et al. construct the following constrained relations, which have a shared root at $(d, p + q - 1)$:

$$
N - 1 \equiv 0 \pmod{g}
$$

$$
e - x_1 \equiv 0 \pmod{g}
$$

$$
N - x_2 \equiv 0 \pmod{g^2}.
$$

We ran our shift polynomial strategies on the Common Prime RSA problem and report the results in Table [22.](#page-44-0)

Table 22: Common Prime RSA. Largest solvable bound for 1000-bit modulus N with 450-bit g, which is studied in [\[25\]](#page-29-1). Since the input relations have mixed moduli, we don't run our precomputation strategy.

		$\left[25\right]$			All monoms.			Graph search		Precomputed		
\boldsymbol{k}	$\lg X$	Dim.	Time	$\lg X$		$\rm Dim.$ Time			$\lg X$ Dim. Time			$\lg X$ Dim. Time
1	132	4	0.1	132	4	0.5	132	4	0.1			
$\overline{2}$	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 . . T.	105	90.8 п	317	106 TT	119.8 0.0100	317 \sim	105	115.9 \mathbf{L}	317 TT	111 0.0000	92.5

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

We ran Algorithm [3](#page-23-1) on the Common prime RSA problem with

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

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

and got output

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

This nearly matches the existing asymptotic bounds of $X_1 < N^{4(0.45)^3}$ [\[25\]](#page-29-1).

A.13 Small Secret Exponent with Multi-Power RSA

In 2014, Lu et al. studied attacks on Multi-Power RSA with small secret expo-nents [\[25\]](#page-29-1). 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

$$
ex - 1 \equiv 0 \pmod{p^2}
$$

 $N \equiv 0 \pmod{p^3}$.

We ran our shift polynomial strategies on the small secret exponent multipower RSA problem and report the results in Table [23.](#page-45-0)

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 [\[25\]](#page-29-1). Since the input relations have mixed moduli, we don't run our precomputation strategy.

		[25]		All monoms.			Graph search				\parallel Precomputed		
\boldsymbol{k}	$\lg X$					$ \text{Dim.} \text{Time} \text{lg }X \text{Dim.} \text{Time} \text{lg }X \text{Dim.} \text{Time} \text{lg }X \text{Dim.} \text{Time} $							
10	594	9	0.4	594	-9	0.8	594	9	0.8 ₁				
20	680	16	1.1 ¹	680	16	-3.5	680	16	3.2 ₁				
30	706	22	4.41	706	22	15.5	706	22	15.1	704	21	4.1	
40	718	29	11.9	718	29	47.91	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](#page-23-1) 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 $X < p^{3(3-1)/(3+1)}$ [\[25\]](#page-29-1).