# Lattice-based Cryptanalysis Toolkit

Joseph Surin Shaanan Cohney

University of Melbourne

**Abstract.** The applicability of lattice reduction to a wide variety of cryptographic situations makes it an important part of the cryptanalyst's toolbox. Despite this, the construction of lattices and use of lattice reduction algorithms for cryptanalysis continue to be somewhat difficult to understand for beginners. This tutorial aims to be a gentle but detailed introduction to lattice-based cryptanalysis targeted towards the novice cryptanalyst with little to no background in lattices. We explain some popular attacks through a conceptual model that simplifies the various components of a lattice attack.

# Contents

# 1 Introduction

Since its invention in 1982, the LLL algorithm [LLL82], and more generally, *lattice reduction*, has enriched the field of cryptanalysis with its wide applicability to attacking all kinds of public key cryptographic constructions. Among many others, lattice reduction has been used to attack the Merkle-Hellman cryptosystem [Adl83], truncated LCGs [Fri+88], RSA with small public exponent [Cop97], RSA with small private exponent [BV96], and (EC)DSA with bad nonces [HS01].

Although the literature is rich with the applications of lattice reduction to cryptanalysis, the algorithms presented and mathematical background required may often make it difficult to approach. In this tutorial, we will present several lattice-based attacks and the necessary background required to understand and develop such attacks. The intent of this tutorial is to be a light introduction to lattice-based cryptanalysis for the novice cryptanalyst with little to no background in lattices. A high level understanding of how lattices can be used for cryptanalysis, intuitions for when and why certain attacks will work, and motivations with practical examples may set a good foundation for one's further, more rigorous studies on the topic.

The structure of this tutorial follows the anatomy of a lattice-based attack, which we consider as consisting of 4 parts with 3 main steps.

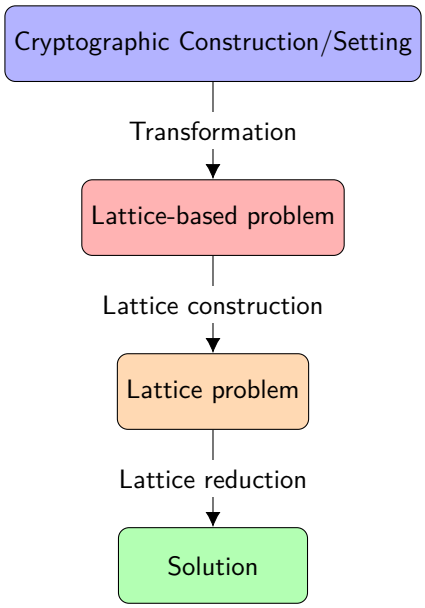



Figure 1: Anatomy of a lattice-based attack.

As in Figure 1, we begin with a cryptographic construction in a particular setting (e.g. ECDSA with biased nonces) and transform it into some lattice-based problem (e.g. Hidden number problem). By constructing a lattice, the lattice-based problem can be reduced to a lattice problem (e.g. $\mathsf{SVP}_\gamma$). Finally, if the bounds allow for it, this lattice problem can be efficiently solved with lattice reduction algorithms to obtain the solution.

Figure 2 shows these steps in context with some concrete examples. In general, a particular lattice-based problem can be the foundation of an attack for many different situations. These lattice-based problems are in turn solved by solving lattice problems, which is where lattice reduction plays a key role.

This ideology will guide the structure of this tutorial. We will start from the foundations by studying the popular lattice reduction algorithm, LLL, and how to use lattice reduction to solve lattice problems. Then, we will look at some lattice-based problems and how to formulate lattices to solve those. Finally, we will study attacks on cryptographic constructions and see how each setting is transformed into a lattice-based problem.

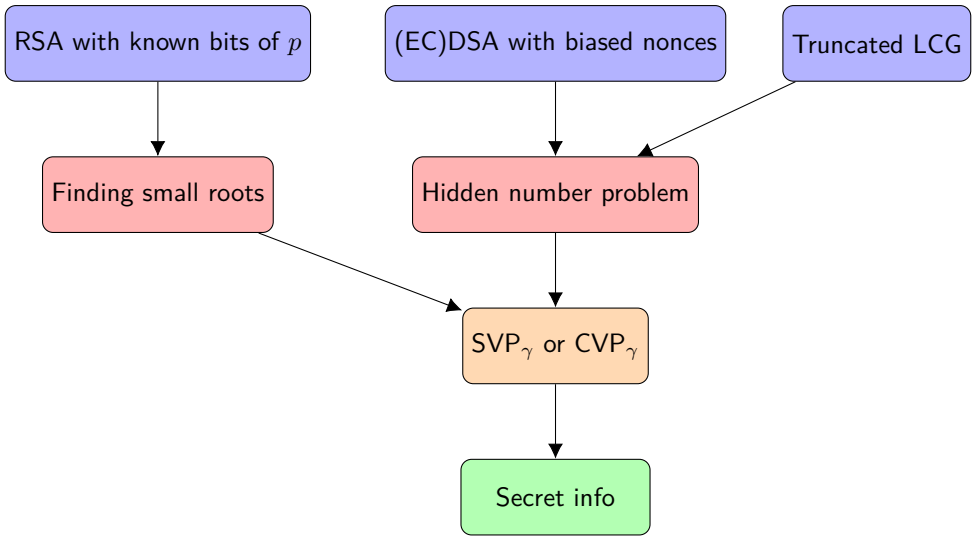



Figure 2: Examples of lattice-based attacks.

Table 1 summarises the lattice-based problems and attacks that we will study in this tutorial.

| Lattice-based Problem | Attack | Description |
|---|---|---|
| Finding small roots | RSA stereotyped message | Low exponent RSA with large amount of known plaintext |
| | Boneh-Durfee attack | RSA with small private exponent $d < 0.292$ |
| | Partial key exposure attack | RSA with small private exponent $d$ and known bits of $d$ |
| Knapsack problem | ECDSA with $k = z \oplus d$ | ECDSA given many signatures calculated with nonce as the message hash XOR the private key |
| Hidden number problem | ECDSA with biased nonces | ECDSA given many signatures calculated with biased nonces |
| | Bleichenbacher's PKCS#1 v1.5 padding oracle attack | PKCS#1 v1.5 given a large number of requests to a decryption padding oracle |
| Extended hidden number problem | ECDSA key disclosure problem | ECDSA given many signatures calculated with partially known nonces and known bits of private key |

Table 1: Attacks studied in this tutorial.

# 2 Background

## 2.1 Notation

Throughout this tutorial, we use bold uppercase letters such as $\mathbf{B}$ to denote matrices and bold lowercase letters such as $\mathbf{b}$ to denote vectors. We denote the length of a vector $\mathbf{v}$ with respect to the Euclidean norm by $\|\mathbf{v}\|$. When describing a lattice or a basis as a matrix, we write the basis vectors as rows of the matrix.

## 2.2 Lattices

### 2.2.1 Basic Definitions

**Definition 2.1 (Lattice).** An *n-dimensional lattice* $\mathcal{L}$ is a discrete, (additive) subgroup of $\mathbb{R}^n$.

A lattice can be described by a *basis* $\mathbf{B}$ which contains linearly independent basis vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$. The number of vectors in the basis, $m$, is called the *rank* of the lattice. In this tutorial, we focus mostly on *full-rank* lattices, which have the same rank and dimension, i.e. $m = n$. The lattice itself can be thought of as the set of all integer linear combinations of these basis vectors:

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{m} a_i \mathbf{b}_i \mid a_i \in \mathbb{Z} \right\}$$

Figure 3 depicts a two dimensional lattice. Note that every non-trivial lattice consists of an infinite number of elements (called lattice points), which are finitely generated by linear combinations of the basis vectors. With this visualisation, we see that a lattice basis is not unique. Figure 3 shows two different bases that generate the same lattice.

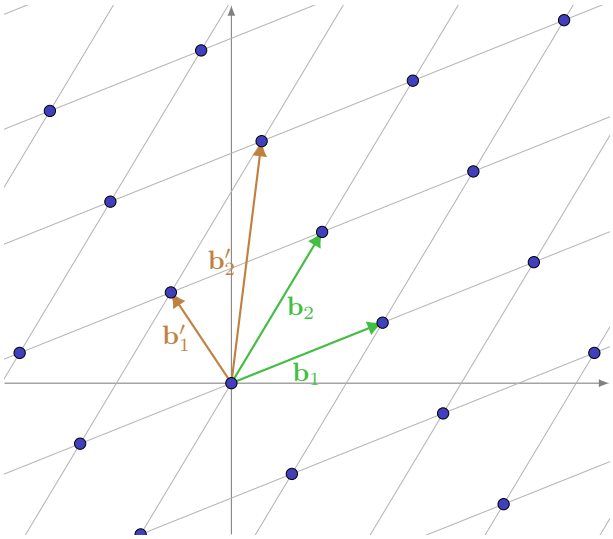



Figure 3: A 2-dimensional lattice and two different bases for it.

In fact, every non-trivial lattice has an infinite number of bases. However, although a lattice can be represented by many different bases, there are some particularly "good" bases that are ideal for solving certain computational problems. The following section defines some properties of lattices which will help us to establish an idea of what constitutes a "good" basis, and eventually, how to find one.

### 2.2.2 Properties, Invariants and Characterisations

Two bases are both bases for the same lattice if they both generate exactly every point in the lattice. But with bases coming in many different shapes and sizes, how can we efficiently determine whether or not two particular bases are bases for the same lattice? To answer this, we will define some properties of lattices bases, as well as some lattice *invariants*: quantities that are intrinsic to the lattice and do not change depending on the basis chosen.

**Definition 2.2 (Fundamental Parallelepiped).** Let $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ be a basis. The *fundamental parallelepiped* of $\mathbf{B}$ is defined as

$$\mathcal{P}(\mathbf{B}) = \left\{ \sum_{i=1}^{m} a_i \mathbf{b}_i \mid a_i \in [0, 1) \right\}$$

Figure 4 shows a lattice and a basis, with the basis' fundamental parallelepiped shaded in. Note that the region is half-open and does not include the three non-zero lattice points near the perimeter.

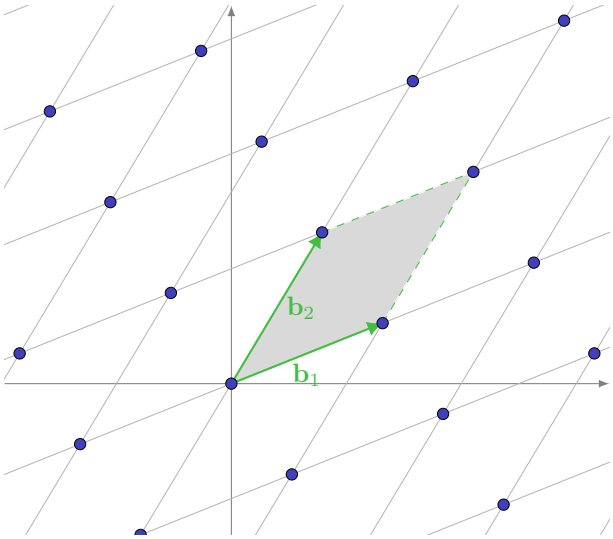



Figure 4: A 2-dimensional lattice with basis $\{\mathbf{b}_1, \mathbf{b}_2\}$ and its associated fundamental parallelepiped.

In fact, one way to characterise lattice bases geometrically is by observing whether or not its fundamental parallelepiped contains a non-zero lattice point. A basis $\mathbf{B}$ is a basis for the lattice $\mathcal{L}$ if and only if $\mathcal{P}(\mathbf{B}) \cap \mathcal{L} = \{\mathbf{0}\}$. The vectors $\{\mathbf{b}_1, \mathbf{b}_2\}$ in figure 5a do not form a basis for the lattice as the fundamental parallelepiped contains a non-zero lattice point. Though we omit a proof here, the two dimensional case provides some intuition as to why this is true: if the fundamental parallelepiped contains a lattice point, then it is impossible to express that point as an integer linear combination of the basis vectors. In the other direction, if the fundamental parallelepiped contains only the zero vector, then we can imagine it as tiling all of $\mathbb{R}^n$ with each parallelepiped containing exactly one lattice point as in figure 5b.

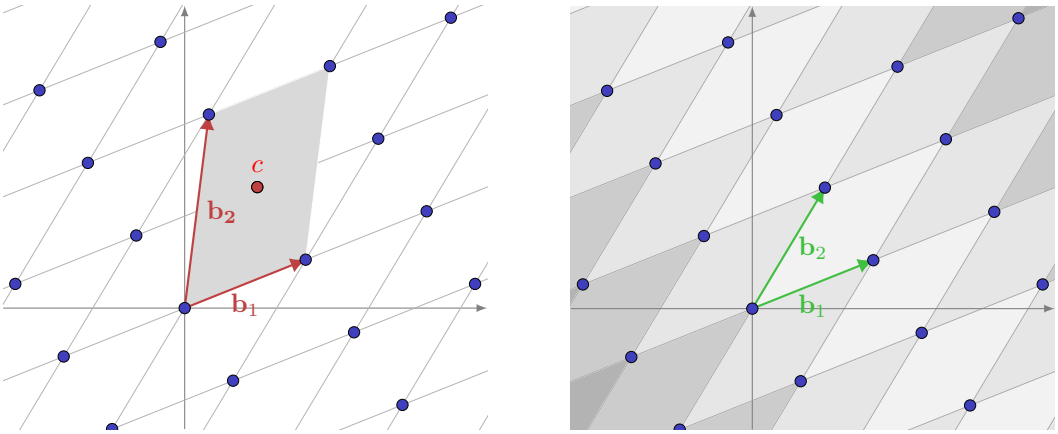
Clearly, a different choice of basis will produce a different fundamental parallelepiped, even if both bases represent the same lattice. However, one invariant quantity is the *volume* of each fundamental parallelepiped, which is the lattice's *determinant*.

**Definition 2.3 (Determinant).** Let $\mathbf{B}$ be a basis for the full-rank lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$. The *determinant* of $\mathcal{L}$, denoted by $\det(\mathcal{L})$, is the $n$-dimensional volume of $\mathcal{P}(\mathbf{B})$. We have

$$\det(\mathcal{L}) = \mathrm{vol}(\mathcal{P}(\mathbf{B})) = |\det(\mathbf{B})|$$

It isn't immediately obvious why the determinant is a lattice invariant. Before we see why it is, we take a detour and give an algebraic characterisation of lattice bases.

Suppose we have two bases $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ and $\mathbf{B}' = \{\mathbf{b}'_1, \ldots, \mathbf{b}'_n\}$ for a full-rank lattice $\mathcal{L}$. Each vector $\mathbf{b}'_1, \ldots, \mathbf{b}'_n$ is in the lattice, so they can be written uniquely as integer linear combinations of the




(a) $\{\mathbf{b}_1, \mathbf{b}_2\}$ do not form a basis, as $\mathcal{P}(\{\mathbf{b}_1, \mathbf{b}_2\})$ contains the non-zero lattice point $c$.

(b) $\{\mathbf{b}_1, \mathbf{b}_2\}$ form a basis, as $\mathcal{P}(\{\mathbf{b}_1, \mathbf{b}_2\})$ contains only the zero point, and tiles $\mathbb{R}^2$.

Figure 5: Geometric characterisation of lattice bases

basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$:

$$\begin{aligned} \mathbf{b}'_1 &= a_{11}\mathbf{b}_1 + \cdots + a_{1n}\mathbf{b}_n \\ &\vdots \\ \mathbf{b}'_n &= a_{n1}\mathbf{b}_1 + \cdots + a_{nn}\mathbf{b}_n \end{aligned}$$

We can write this as the matrix equation $\mathbf{B}' = \mathbf{AB}$ where $\mathbf{A}$ is the coefficient matrix of this system of equations. In the same way, the $\mathbf{b}_i$ can also be written uniquely as integer linear combinations of the $\mathbf{b}'_i$. Specifically, we have $\mathbf{B} = \mathbf{A}^{-1}\mathbf{B}'$. Noting that $\mathbf{A}^{-1}$ must be a matrix of integers, we have

$$\det(\mathbf{A}\mathbf{A}^{-1}) = \det(\mathbf{A})\det(\mathbf{A}^{-1}) = \det(\mathbf{I}) = 1$$

with $\det(\mathbf{A})$ and $\det(\mathbf{A}^{-1})$ being integers. Therefore, we must have $|\det(\mathbf{A})| = 1$. Conversely, suppose that for some integer matrix $\mathbf{A}$ with $|\det(\mathbf{A})| = 1$ we have $\mathbf{B} = \mathbf{AB}'$. In this case, clearly each vector in $\mathbf{B}$ can be written as an integer linear combination of the vectors in $\mathbf{B}'$, so $\mathcal{L}(\mathbf{B}) \subseteq \mathcal{L}(\mathbf{B}')$. We also have $\mathbf{B}' = \mathbf{A}^{-1}\mathbf{B}$, so by the same argument, we have $\mathcal{L}(\mathbf{B}') \subseteq \mathcal{L}(\mathbf{B})$. Therefore, $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$. This leads us to the following theorem which will play a key role in the following sections.

**Theorem 2.4.** Let $\mathbf{B}$ and $\mathbf{B}'$ be bases. Then, $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$ if and only if $\mathbf{B} = \mathbf{UB}'$ for some integer matrix $\mathbf{U}$ with $|\det(\mathbf{U})| = 1$.

With this theorem, it is easy to see why the determinant of a lattice is invariant: suppose $\mathbf{B}$ and $\mathbf{B}'$ are bases of the lattice $\mathcal{L}$. Then $\mathbf{B} = \mathbf{UB}'$, so $\det(\mathbf{B}) = \det(\mathbf{UB}') = \det(\mathbf{U})\det(\mathbf{B}') = \det(\mathbf{B}')$. The determinant will be a useful property for analysing algorithms and bounds later on.

Another important invariant of a lattice is its successive minima. The first successive minimum, often denoted as $\lambda(\mathcal{L})$ or just $\lambda$ (when the context is clear), is the length of the shortest non-zero vector in the lattice. In general, a lattice of rank $n$ has $n$ successive minima which are defined as follows.

**Definition 2.5 (Successive Minima).** Let $\mathcal{L}$ be a lattice of rank $n$. For $i \in \{1, \ldots, n\}$, the $i$th successive minimum of $\mathcal{L}$, denoted by $\lambda_i(\mathcal{L})$, is the smallest $r$ such that $\mathcal{L}$ has $i$ linearly independent vectors of length at most $r$.

Geometrically, $\lambda_i(\mathcal{L})$ is the radius of the smallest closed ball around the origin which contains $i$ linearly independent vectors. Figure 6 depicts the first successive minimum of a lattice.

Interestingly, there is no efficient algorithm to compute the successive minima of a lattice. There is, however, an upper bound due to Minkowski.

**Theorem 2.6 (Minkowski's First Theorem).** Let $\mathcal{L}$ be a full-rank $n$ dimensional lattice. Then

$$\lambda_1(\mathcal{L}) \leq \sqrt{n}|\det(\mathcal{L})|^{1/n}$$

The first successive minimum is of particular interest as it gives a standard by which we can judge the length of vectors in a lattice.

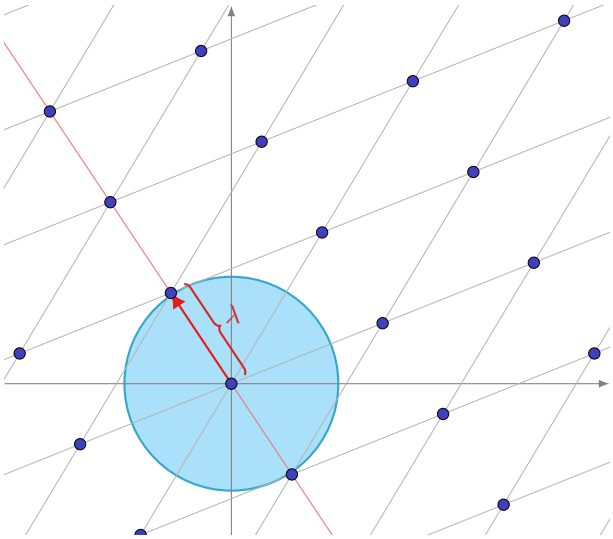



Figure 6: First successive minimum $\lambda$ and a shortest non-zero vector of a lattice. The red line is the one dimensional span of the shortest vectors.

## 2.3 Lattice Problems

We now define some important computational *lattice problems*.

**Definition 2.7 (Shortest Vector Problem (SVP)).** Given a basis $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, find a non-zero lattice vector $\mathbf{v}$ that satisfies $\|\mathbf{v}\| = \lambda_1(\mathcal{L})$.

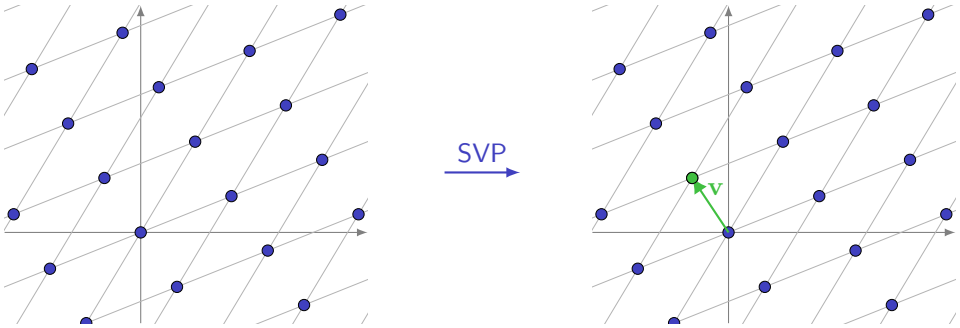



Figure 7: The Shortest Vector Problem.

**Definition 2.8 (Closest Vector Problem (CVP)).** Given a basis $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, and a target vector $\mathbf{t}$ (not necessarily in $\mathcal{L}$), find a lattice vector $\mathbf{v}$ that satisfies $\|\mathbf{v} - \mathbf{t}\| = \min_{\mathbf{w} \in \mathcal{L}} \|\mathbf{w} - \mathbf{t}\|$.

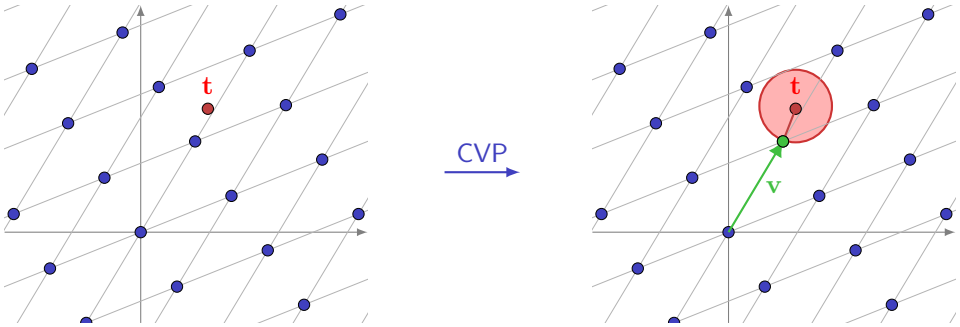



Figure 8: The Closest Vector Problem.

It is well-known that these problems are NP-Hard [Ajt98]. However, we can define slightly relaxed

versions of these problems which turn out to have efficient solutions for certain parameters. These will be more useful for cryptanalysis.

**Definition 2.9 (Approximate Shortest Vector Problem (SVP$_\gamma$)).** Given a basis $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, find a non-zero lattice vector $\mathbf{v}$ that satisfies $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\mathcal{L})$.

**Definition 2.10 (Approximate Closest Vector Problem (CVP$_\gamma$)).** Given a basis $\mathbf{B}$ of a lattice $\mathcal{L} = \mathcal{L}(\mathbf{B})$, and a target vector $\mathbf{t}$, find a lattice vector $\mathbf{v}$ that satisfies $\|\mathbf{v} - \mathbf{t}\| \leq \gamma \cdot \min_{\mathbf{w} \in \mathcal{L}} \|\mathbf{w} - \mathbf{t}\|$.

To solve these computational problems, we use *lattice reduction*. The idea is that we can transform an arbitrary lattice basis into a "better" basis that contains shorter and more orthogonal vectors. In the next section, we describe the LLL algorithm [LLL82] which is a lattice reduction algorithm that gives a polynomial-time solution to these problems for approximation factors exponential in the lattice dimension.

## 2.4 Lattice Reduction

The goal of lattice reduction is to take an arbitrary lattice basis and transform it into another basis for the same lattice that has shorter and more orthogonal vectors. Since we look for short vectors, this process in itself may yield a solution for the approximate shortest vector problem.

Recall from Theorem 2.4 that two bases $\mathbf{B}$ and $\mathbf{B}'$ represent the same lattice if and only if $\mathbf{B} = \mathbf{U}\mathbf{B}'$ for some integer matrix $\mathbf{U}$ with $|\det(\mathbf{U})| = 1$. This leads us to two useful transformations we can perform on a lattice basis: vector-switching and vector-addition. Left multiplying a basis by the matrix $T_{i,j}$ yields a new basis with the $i$th and $j$th basis vectors swapped, and left multiplying a basis by the matrix $L_{i,j}(k)$ yields a new basis with the $j$th basis vector added $k$ times to the $i$th basis vector. Note that both of these matrices have determinants of $\pm 1$. These two elementary row operations are key parts of the LLL algorithm.

$$T_{i,j} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & 1 & & \\ & & & \ddots & & & \\ & & 1 & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ i \\ \\ j \\ \\ \end{matrix}$$

$$L_{i,j}(k) = \begin{bmatrix} & & & j & & & \\ 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & k & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ i \\ \\ \end{matrix}$$

### 2.4.1 The LLL Algorithm

We now describe the algorithm of Lenstra, Lenstra and Lovász [LLL82].

The first step of this iterative algorithm is to take the basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ and compute an orthogonal basis by the Gram-Schmidt orthogonalisation process. This orthogonal basis is not a valid basis for the lattice but will help us to reduce the lattice basis. The Gram-Schmidt process gives us the orthogonal vectors $\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$ and coefficients $\mu_{i,j}$ defined as

$$\begin{cases} \mathbf{b}_i^* = \mathbf{b}_i, & i = 1 \\ \mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}_j^*, & 1 < i \leq n \end{cases} \qquad \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$$

With this, we have the notion of a LLL-reduced basis which is our end goal.

**Definition 2.11 ($\delta$-LLL reduced).** Let $\delta \in (1/4, 1)$. A basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ is $\delta$-LLL-reduced if

1. $|\mu_{i,j}| \leq 1/2$ for all $i > j$ **(size-reduced)**
2. $(\delta - \mu_{i+1,i}^2)\|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2$ for all $1 \leq i \leq n-1$ **(Lovász condition)**

The first condition relates to the lengths of the basis vectors but it isn't sufficient to ensure all basis vectors are short (consider the two dimensional case where $\mathbf{b}_1$ is large). The second condition helps to rectify this by imposing a requirement on a more local scale between consecutive Gram-Schmidt vectors which roughly says that the second vector should be not much shorter than the first.

With these definitions, we can now present the LLL algorithm.

**Algorithm 1** LLL Algorithm [LLL82]

```
function LLL(Basis {b_1, ..., b_n}, δ)
    while true do
        for i = 2 to n do                                   ▷ size-reduction
            for j = i − 1 to 1 do
                b_i^*, μ_{i,j} ← Gram-Schmidt(b_1, ..., b_n)
                b_i ← b_i − ⌊μ_{i,j}⌉ b_j
        if ∃i such that (δ − μ_{i+1,i}^2)‖b_i^*‖^2 > ‖b_{i+1}^*‖^2 then   ▷ Lovász condition
            Swap b_i and b_{i+1}
        else
            return {b_1, ..., b_n}
```

**Example 2.12.** Let $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2\}$ where $\mathbf{b}_1 = (-2, 2), \mathbf{b}_2 = (-2, 1)$. We will run the LLL algorithm on this lattice basis, using $\delta = 0.75$

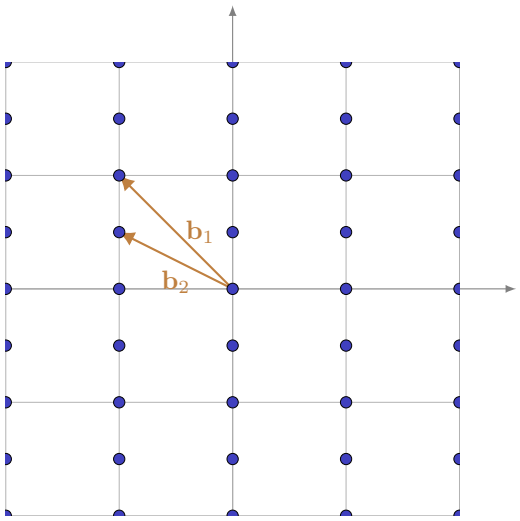



Figure 9: Lattice $\mathcal{L}(\mathbf{B})$ with basis $\mathbf{b}_1 = (-2, 2)$ and $\mathbf{b}_2 = (-2, 1)$.

We begin by computing the first set of Gram-Schmidt vectors and coefficients:

$$\begin{aligned}
\mathbf{b}_1^* &= \mathbf{b}_1 && = (-2, 2) \\
\mathbf{b}_2^* &= \mathbf{b}_2 - \frac{\langle \mathbf{b}_2, \mathbf{b}_1^* \rangle}{\langle \mathbf{b}_1^*, \mathbf{b}_1^* \rangle} \mathbf{b}_1^* = (-2, 1) - 0.75 \cdot (-2, 2) && = (-0.5, -0.5) \\
\mu_{2,1} &= \frac{\langle \mathbf{b}_2, \mathbf{b}_1^* \rangle}{\langle \mathbf{b}_1^*, \mathbf{b}_1^* \rangle} && = 0.75
\end{aligned}$$

Then, we set $\mathbf{b}_2 \leftarrow \mathbf{b}_2 - \lfloor \mu_{2,1} \rceil \mathbf{b}_1 = (0, -1)$.

The Lovász condition between $\mathbf{b}_1^*$ and $\mathbf{b}_2^*$ is not satisfied as

$$(\delta - \mu_{2,1}^2)\|\mathbf{b}_1^*\|^2 = (0.75 - 0.75^2) \cdot 8 = 1.5 > 0.5 = \|\mathbf{b}_2^*\|^2$$

so we swap vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ in the basis. We now have $\mathbf{b}_1 = (0, -1)$ and $\mathbf{b}_2 = (-2, 2)$.

We perform another iteration and compute the Gram-Schmidt vectors and coefficients from the new basis:

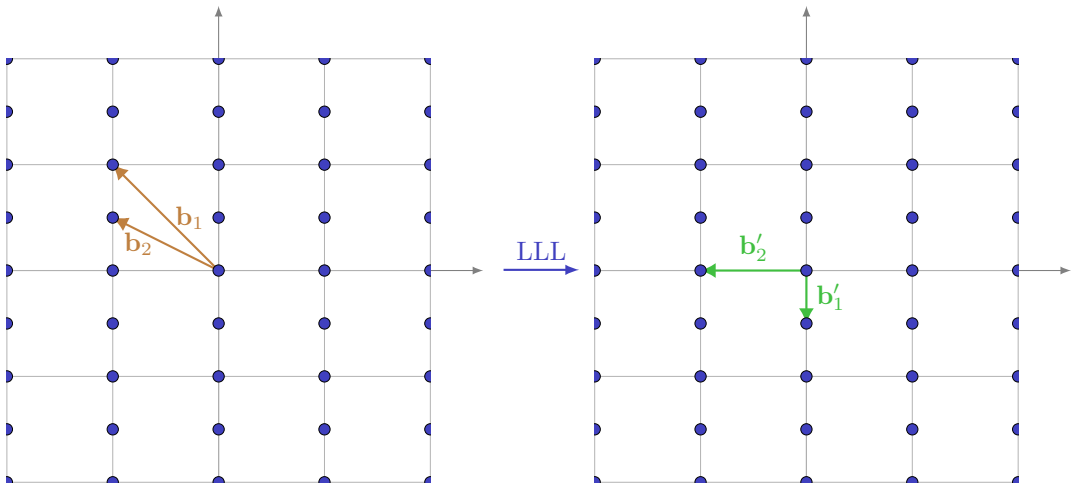
$$\begin{aligned}
\mathbf{b}_1^* &= \mathbf{b}_1 && = (0, -1) \\
\mathbf{b}_2^* &= \mathbf{b}_2 - \frac{\langle \mathbf{b}_2, \mathbf{b}_1^* \rangle}{\langle \mathbf{b}_1^*, \mathbf{b}_1^* \rangle} \mathbf{b}_1^* = (-2, 2) + 2 \cdot (0, -1) && = (-2, 0) \\
\mu_{2,1} &= \frac{\langle \mathbf{b}_2, \mathbf{b}_1^* \rangle}{\langle \mathbf{b}_1^*, \mathbf{b}_1^* \rangle} && = -2
\end{aligned}$$

We find that the Lovász condition holds (for every pair of consecutive Gram-Schmidt vectors):

$$(\delta - \mu_{2,1}^2)\|\mathbf{b}_1^*\|^2 = (0.75 - (-2)^2) \cdot 1 = -3.25 \leq 4 = \|\mathbf{b}_2^*\|^2$$

so we are done.

The LLL-reduced basis is $\{\mathbf{b}_1', \mathbf{b}_2'\}$ where $\mathbf{b}_1' = (0, -1)$ and $\mathbf{b}_2' = (-2, 0)$.




(a) Lattice $\mathcal{L}(\mathbf{B})$ with original basis $\mathbf{b}_1 = (-2, 2)$, $\mathbf{b}_2 = (-2, 1)$.

(b) Lattice with LLL-reduced basis $\mathbf{b}_1' = (0, -1), \mathbf{b}_2' = (-2, 0)$.

We note that in this toy example, a shortest vector in the lattice is found in the first vector of the reduced basis!

Of course, LLL does not always find a shortest vector. But we can show that it will always find a vector within an approximation factor exponential in the dimension of the lattice to the shortest vector. To do so, we first prove a theorem for the lower bound on the shortest vector in a lattice.

**Theorem 2.13.** Let $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be a lattice basis and $\mathbf{B}^* = \{\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*\}$ its corresponding Gram-Schmidt orthogonalisation. Then $\lambda_1(\mathcal{L}(\mathbf{B})) \geq \min_{i \in \{1,\ldots,n\}} \|\mathbf{b}_i^*\|$.

*Proof.* Let $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}^n$ be a non-zero vector. We consider the lattice point $\mathbf{xB}$ and show that its length is bounded below by $\min_{i \in \{1,\ldots,n\}} \|\mathbf{b}_i^*\|$. Let $j$ be the largest index such that $x_j \neq 0$. Then

$$\begin{aligned}
|\langle \mathbf{xB}, \mathbf{b}_j^* \rangle| &= |\langle \sum_{i=1}^{n} x_i \mathbf{b}_i, \mathbf{b}_j^* \rangle| \\
&= |\sum_{i=1}^{n} x_i \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle| && \text{by linearity of the inner product} \\
&= |x_j| \langle \mathbf{b}_j, \mathbf{b}_j^* \rangle && \text{since } \mathbf{b}_i \text{ and } \mathbf{b}_j^* \text{ are orthogonal for } j < i \text{ and } x_i = 0 \text{ for } i < j \\
&= |x_j| \cdot \|\mathbf{b}_j^*\|^2
\end{aligned}$$

From the Cauchy-Schwartz inequality, we have

$$\begin{aligned} |\langle \mathbf{xB}, \mathbf{b}_j^* \rangle|^2 &\leq \langle \mathbf{xB}, \mathbf{xB} \rangle \cdot \langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle \\ \implies |\langle \mathbf{xB}, \mathbf{b}_j^* \rangle| &\leq \|\mathbf{xB}\| \cdot \|\mathbf{b}_j^*\| \end{aligned}$$

so

$$\begin{aligned} |x_j| \cdot \|\mathbf{b}_j^*\|^2 &\leq \|\mathbf{xB}\| \cdot \|\mathbf{b}_j^*\| \\ \implies |x_j| \cdot \|\mathbf{b}_j^*\| &\leq \|\mathbf{xB}\| \\ \implies \|\mathbf{b}_j^*\| &\leq \|\mathbf{xB}\| \qquad \text{since } x_j \neq 0 \text{ is an integer} \\ \implies \min_{i \in \{1,\ldots,n\}} \|\mathbf{b}_i^*\| &\leq \|\mathbf{xB}\| \end{aligned}$$

Therefore, since $\mathbf{xB}$ can be any non-zero lattice point, we must have $\lambda_1(\mathcal{L}(\mathbf{B})) \geq \min_{i \in \{1,\ldots,n\}} \|\mathbf{b}_i^*\|$. □

**Proposition 2.14.** Let $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be a $\delta$-LLL-reduced basis. Then $\|\mathbf{b}_1\| \leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{n-1} \lambda_1$.

*Proof.* From the Lovász condition, we have

$$(\delta - \mu_{i+1,i}^2) \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2$$

and from the size-reduced condition, we have $|\mu_{i+1,i}| \leq \frac{1}{2}$, so

$$\left(\frac{4\delta - 1}{4}\right) \|\mathbf{b}_i^*\|^2 \leq \|\mathbf{b}_{i+1}^*\|^2$$

By chaining the inequalities, we get

$$\begin{aligned} \|\mathbf{b}_1^*\|^2 = \|\mathbf{b}_1\|^2 &\leq \left(\frac{4}{4\delta - 1}\right)^{i-1} \|\mathbf{b}_i^*\|^2 \\ \implies \|\mathbf{b}_1\| &\leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{i-1} \|\mathbf{b}_i^*\| \\ \implies \|\mathbf{b}_1\| &\leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{n-1} \|\mathbf{b}_n^*\| \end{aligned}$$

The result then follows from Lemma 2.13. □

Combining Proposition 2.14 with Theorem 2.6, we have the result

**Proposition 2.15.**

$$\|\mathbf{b}_1\| \leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{n-1} \sqrt{n} \cdot |\det(\mathcal{L})|^{1/n}$$

This result will be important for us as it gives us a rough idea about what short vectors LLL will find. As we'll see throughout this tutorial, if we can model a particular problem in which the solution exists as a short vector of some lattice, we may be able to solve for it by using lattice reduction if the short vector is within an exponential approximation factor of the shortest vector. We omit the analysis and proof and take it for granted that the LLL algorithm runs in time polynomial in the lattice dimension.

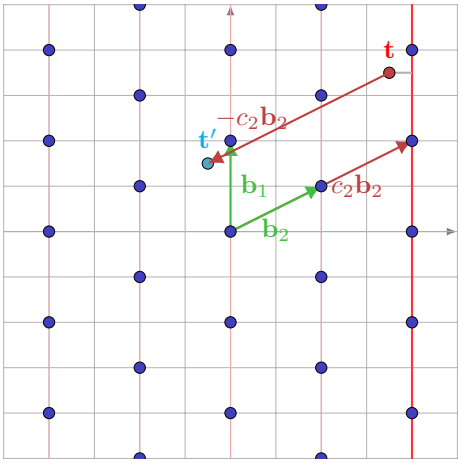
## 2.5 Solving CVP

As we have seen, the LLL lattice reduction algorithm can be used to solve the approximate shortest vector problem with approximation factors by simply taking the first vector in the LLL-reduced basis. Lattice reduction can also be used to solve the approximate closest vector problem with similar approximation factors. In this section, we briefly discuss Babai's nearest plane algorithm [Bab86] and Kannan's embedding method [Kan87].

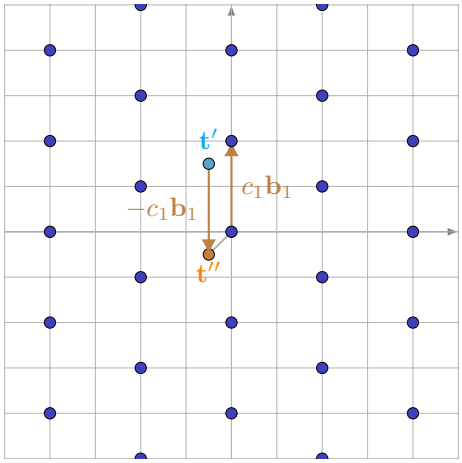
### 2.5.1 Babai's Nearest Plane Algorithm

Babai's nearest plane algorithm is a greedy algorithm that uses induction on the dimension $n$ of the lattice. The algorithm begins with lattice reduction to get a reduced basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$. Where $\mathbf{t}$ is the target vector, we consider the hyperplane generated by the first $n-1$ lattice vectors and let $\mathbf{t}' = \mathbf{t} - c_n\mathbf{b}_n$ where $c_n$ is an integer such that the hyperplane translated by $c_n\mathbf{b}_n$ is as close as possible to $\mathbf{t}$. $c_n$ can be computed as $c_n = \lceil \langle \mathbf{t}, \mathbf{b}_n^* \rangle / \langle \mathbf{b}_n^*, \mathbf{b}_n^* \rangle \rfloor$. We then inductively apply this process to the first $n-1$ lattice vectors and the new translated target vector $\mathbf{t}'$. The output of the algorithm is the sum of the $c_i\mathbf{b}_i$ which is clearly a lattice point.

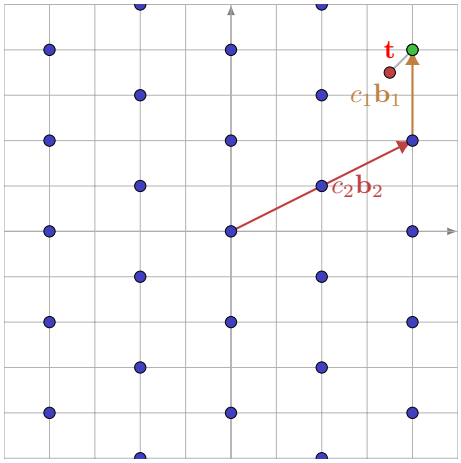
Figure 11 gives an example of the algorithm in the two dimensional case.




(a) The red lines represent the translated hyperplanes $z\mathbf{b}_2 + \text{span}\{\mathbf{b}_1\}$ with $z \in \mathbb{Z}$. $c_2\mathbf{b}_2 + \text{span}\{\mathbf{b}_1\}$ (drawn in bold) is the closest translated hyperplane to the target $\mathbf{t}$.




(b) The hyperplane in this iteration is $\{\mathbf{0}\}$, so we find the multiple $c_1$ of $\mathbf{b}_1$ that minimises the length of $\mathbf{t}' - c_1\mathbf{b}_1$.




(c) The closest lattice point to $\mathbf{t}$ is given by $c_2\mathbf{b}_2 + c_1\mathbf{b}_1$.

Figure 11: Babai's Nearest Plane Algorithm.

The time complexity of Babai's algorithm is dominated by the lattice reduction step. Thus, in the case where we use LLL, the algorithm runs in time polynomial in the lattice dimension. As one might expect, when using Babai's algorithm with LLL, we can solve $\mathsf{CVP}_\gamma$ to within an exponential approximation factor.

**Algorithm 2** Babai's Nearest Plane Algorithm [Bab86]

**function** BABAI(Basis $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$, target vector $\mathbf{t}$)
  Perform lattice reduction on $\mathbf{B}$
  $\mathbf{b}_i^* \leftarrow$ Gram-Schmidt$(\mathbf{b}_1, \ldots, \mathbf{b}_n)$
  $\mathbf{b} \leftarrow \mathbf{t}$
  **for** $i = n - 1$ to $1$ **do**
    $c_i \leftarrow \lceil \langle \mathbf{b}, \mathbf{b}_i^* \rangle / \langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle \rfloor$
    $\mathbf{b} \leftarrow \mathbf{b} - c_i \mathbf{b}_i$
  **return** $\mathbf{t} - \mathbf{b}$

### 2.5.2 Kannan's Embedding Method

Kannan's embedding method is another technique to solve the closest vector problem which works by embedding the target vector in the lattice basis and treating the CVP instance as a SVP instance. Let $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ be the lattice basis and let $\mathbf{t} = (t_1, \ldots, t_n)$ be the target vector. Suppose that a solution to the CVP instance is given by $c_1\mathbf{b}_1 + \cdots + c_n\mathbf{b}_n$. Then, we have

$$\mathbf{t} \approx \sum_{i=1}^{n} c_i \mathbf{b}_i$$
$$\implies \mathbf{t} = \sum_{i=1}^{n} c_i \mathbf{b}_i + \mathbf{e}$$

where $\|\mathbf{e}\|$ is small. Hence, it makes sense to consider the $n+1$ dimensional lattice with basis

$$\mathbf{B}' = \begin{bmatrix} \mathbf{B} & 0 \\ \mathbf{t} & q \end{bmatrix}$$

which contains the short vector $(\mathbf{e}, q)$ by the linear combination $(-c_1, \ldots, -c_n, 1)$. The solution is then given by subtracting $\mathbf{e}$ from $\mathbf{t}$.

The integer $q$ is known as the embedding factor and often affects how successful LLL will be in revealing the correct vector. We refer to [Gal12] and [Sun+21] for discussions on the embedding factor.

## 2.6 An Application of Lattice Reduction

Before we begin discussing the application of lattice reduction to cryptanalysis, we will look at an application of lattice reduction in algebra. The problem we look at here is that of reconstructing the minimal polynomial of an algebraic number given an approximation of the number. This problem was shown to be solvable with lattice reduction in [KLL84].

**Definition 2.16 (Minimal Polynomial).** Let $\alpha \in F$ where $F$ is a field. The *minimal polynomial* of $\alpha$ is the monic polynomial of lowest degree in $F[x]$ such that $\alpha$ is a root.

We will use lattice reduction to find the minimal polynomial $f(x)$ of $\alpha = 7 + \sqrt[3]{5}$ given the approximation $\beta = 7 + \sqrt[3]{5} \approx 8.70997594$ to 8 decimal places. Firstly, we suspect that $f$ is of degree 3, so we write $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ with $a_i \in \mathbb{R}$. We have $f(\alpha) = 0$, and so $f(\beta) \approx 0$. To work over the integers, we multiply this expression by $10^8$ to get

$$10^8 a_0 + \lfloor 10^8 \beta \rfloor a_1 + \lfloor 10^8 \beta^2 \rfloor a_2 + \lfloor 10^8 \beta^3 \rfloor a_3 \approx 0$$

Now, consider the lattice with basis (given by the rows)

$$\mathbf{B} = \begin{bmatrix} 10^8 & 1 & 0 & 0 \\ \lfloor 10^8 \beta \rfloor & 0 & 1 & 0 \\ \lfloor 10^8 \beta^2 \rfloor & 0 & 0 & 1 \\ \lfloor 10^8 \beta^3 \rfloor & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 100000000 & 1 & 0 & 0 \\ 870997594 & 0 & 1 & 0 \\ 7586368087 & 0 & 0 & 1 \\ 66077083514 & 0 & 0 & 0 \end{bmatrix}$$

Note that our target vector $\mathbf{x} = (c, a_0, a_1, a_2)$ containing the coefficients of the minimal polynomial is in this lattice with the linear combination $\mathbf{t} = (a_0, a_1, a_2, 1)$. That is, $\mathbf{tB} = \mathbf{x}$. Here, $c$ is an integer very close to 0.

To justify why LLL will be successful in helping us to find the coefficients of the minimal polynomial, we assume knowledge of an upper bound, $M = 400$, for the *height* of the minimal polynomial, which is defined to be the maximum of the magnitudes of its coefficients. This gives us a rough upper bound of our target vector: $\|(0, M, M, M)\| \approx 692$. We use the result of the previous section which bounds the first vector in the LLL-reduced basis to find that

$$\|\mathbf{b}_1\| \leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^{n-1} \sqrt{n} |\det(\mathcal{L})|^{1/n} \approx 2868$$

which suggests we should be able to find the target vector with LLL.

In practice, LLL will likely produce a vector much shorter than this upper bound. [NS06] gives a heuristic that suggests the first vector in the LLL-reduced basis will satisfy $\|\mathbf{b}_1\| / |\det(\mathcal{L})|^{1/n} \approx 1.02^n$ for random bases. In general, we will tend to rely on heuristics and rough calculations to determine whether or not LLL will be successful.

Running the LLL algorithm on the basis gives the reduced basis $\mathbf{B}'$

$$\mathbf{B}' = \begin{bmatrix} 5 & -348 & 147 & -21 \\ 438 & -75 & 116 & 188 \\ 109 & -214 & -563 & -159 \\ 357 & 136 & 220 & -419 \end{bmatrix}$$

The first row gives the shortest vector, and we recognise it as our solution because the first entry, 5, is close to 0. We read the coefficients off this vector to get $a_0 = -348, a_1 = 147, a_2 = -21$. So, the minimal polynomial of $\alpha = 7 + \sqrt[3]{5}$ is

$$f(x) = x^3 - 21x^2 + 147x - 348$$

# 3 Lattice-based Problems

In this section, we study some problems which can be solved by modelling the situation as a $\mathsf{SVP}_\gamma$ or $\mathsf{CVP}_\gamma$ instance. These problems may be characterised by their linear nature and throughout this tutorial, we call these problems *lattice-based problems*.

## 3.1 Finding Small Roots

We start with arguably one of the most influential and widely applied problems in public key cryptanalysis: the problem of finding "small" roots of a polynomial modulo an integer. This problem was studied in a seminal work by Coppersmith where he also applied his techniques to breaking low exponent RSA in particular settings [Cop96]. The power of Coppersmith's method is in its ability to find small roots of polynomial equations modulo composite numbers, without knowledge of the factorisation. Finding roots modulo a prime is easy, while finding roots modulo a composite number $N$ is equivalent to factoring $N$, so Coppersmith's method is a good compromise and has indeed found innumerable applications in cryptanalysis.

In this section, we present a simple, informal overview of Coppersmith's method in the univariate case, as well as some useful extensions and generalisations which we state without proof.

### 3.1.1 Coppersmith's Method: An Overview

Let $N$ be a composite integer and $f(x) = x^d + \sum_{i=0}^{d-1} a_i x^i \in \mathbb{Z}[x]$ a monic polynomial of degree $d$. Coppersmith's method helps us to find all integer solutions $x_0$ to the equation $f(x_0) = 0 \pmod N$ which satisfy $|x_0| < B$ for some bound $B$ depending on $N$ and $d$.

The main idea of the algorithm is to construct a polynomial $h(x)$ over the integers which also satisfies $h(x_0) = 0$. Since there are efficient algorithms to find roots of univariate polynomials over the integers, finding such a polynomial will give us the roots to the original polynomial. To proceed, we note that adding integer multiples of $g_i(x) = Nx^i \in \mathbb{Z}[x]$ to $f$ results in polynomials which have the same roots as $f$ modulo $N$. So we consider the lattice generated by the rows of $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} N & & & & & \\ & BN & & & & \\ & & B^2N & & & \\ & & & \ddots & & \\ & & & & B^{d-1}N & \\ a_0 & a_1B & a_2B^2 & \cdots & a_{d-1}B^{d-1} & B^d \end{bmatrix}$$

The first $d$ rows encode the polynomials $g_i(Bx)$ for $0 \leq i < d$ while the last row encodes $f(Bx)$. Thus, every element of this lattice represents a polynomial which shares the roots of $f$ modulo $N$. Crucially, we observe that if there is such a polynomial $h(x)$ in this lattice which also satisfies $|h(x_0)| < N$, then we have $h(x_0) = 0$ over the integers. This polynomial should have "small" coefficients to satisfy this condition. Specifically we require $|h_i x_0^i| \leq |h_i B^i| < N/(d+1)$ for all coefficients $h_i$ of $h$. Since the elements of $\mathcal{L}(\mathbf{B})$ encode the coefficients of these polynomials of interest, perhaps using lattice reduction to find a short vector will be helpful.

$\mathbf{B}$ is triangular, so we can easily calculate $\det(\mathcal{L}(\mathbf{B})) = \det(\mathbf{B}) = B^{d(d+1)/2}N^d$. From Proposition 2.15, we have (using $\delta = 3/4$)

$$\begin{aligned} \|\mathbf{b}_1\| &\leq \left(\frac{2}{\sqrt{4\delta - 1}}\right)^d \sqrt{d+1}\,|\det(\mathcal{L})|^{1/(d+1)} \\ &= 2^{d/2}\sqrt{d+1} \cdot B^{d/2}N^{d/(d+1)} \\ &= 2^{d/2}\sqrt{d+1} \cdot B^{d/2}NN^{-1/(d+1)} \end{aligned}$$

where $\mathbf{b}_1 = (b_0, \ldots, b_d)$ is the first vector in the LLL-reduced basis. We interpret this vector as the coefficients of the polynomial $h(Bx)$, and we see that by setting

$$B < N^{2/d(d+1)}/(2(d+1)^{3/d})$$

we achieve the required bounds on the $h_i$:

$$|h_i B^i| = |b_i| \leq \|\mathbf{b}_1\| < N/(d+1)$$

So, to find the small roots of $f \mod N$, we take the polynomial $h(x) = b_0 + (b_1/B)x + (b_2/B^2)x^2 + \cdots + (b_{d-1}/B^{d-1})x^{d-1} + x_d$ and solve for its roots over the integers. We check each root to ensure that it is a root of $f \mod N$.

**Example 3.1.** Let $N = 23 \cdot 29 = 667$ and $f(x) = x^2 + 6x + 352 \in \mathbb{Z}[x]$. Note that $f$ has the "small" root $x_0 = 15$ modulo $N$ but not over the integers. That is, $f(15) = 0 \pmod N$, but $f(15) \neq 0$. We will use Coppersmith's method as described above to recover this small root. We take $B = 20$ and construct the lattice generated by the rows of $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} N & & & & & \\ & BN & & & & \\ & & B^2N & & & \\ & & & \ddots & & \\ & & & & B^{d-1}N & \\ a_0 & a_1B & a_2B^2 & \cdots & a_{d-1}B^{d-1} & B^d \end{bmatrix} = \begin{bmatrix} 667 & 0 & 0 \\ 0 & 20 \cdot 667 & 0 \\ 352 & 6 \cdot 20 & 20^2 \end{bmatrix} = \begin{bmatrix} 667 & 0 & 0 \\ 0 & 13340 & 0 \\ 352 & 120 & 400 \end{bmatrix}$$

Running LLL on this basis yields the reduced basis $\mathbf{B}'$:

$$\mathbf{B}' = \begin{bmatrix} -315 & 120 & 400 \\ 352 & 120 & 400 \\ 167 & 12260 & -3600 \end{bmatrix}$$

We read off the first row and interpret it as the coefficients of the polynomial $h(Bx)$. So we compute $h(x)$ as follows:

$$\begin{aligned} h(Bx) &= 400x^2 + 120x - 315 \\ \implies h(x) &= \left(\frac{400}{20^2}\right)x^2 + \left(\frac{120}{20}\right)x - 315 \\ &= x^2 + 6x - 315 \end{aligned}$$

Then, solving for the roots of $h(x)$ (i.e. with the quadratic formula or Newton's method), we find the solution $x_0 = 15$. Note that this also gives us the solution $x_0 = -21$ whose magnitude is greater than $B$. In practice, we can often expect Coppersmith's method to perform slightly better than the theoretical bounds.

### 3.1.2 Coppersmith's Method: Extensions and Generalisations

From further work of Coppersmith and many others, Coppersmith's method has been developed into stronger and more useful results. We give a formal statement of Coppersmith's method and some useful extensions and generalisations.

**Theorem 3.2 (Coppersmith's Method).** Let $N$ be an integer of unknown factorisation, which has a divisor $b \geq N^\beta$. Let $f(x)$ be a univariate, monic polynomial of degree $\delta$ and $0 < \epsilon \leq \frac{1}{7}\beta$. Then we can find all solutions $x_0$ of $f(x) = 0 \pmod b$ with

$$|x_0| \leq \frac{1}{2}N^{\frac{\beta^2}{\delta} - \epsilon}$$

in time polynomial in $(\log N, \delta, \frac{1}{\epsilon})$.

**Theorem 3.3 (Coppersmith's Method for Bivariate Integer Polynomials [Cor07]).** Let $f(x, y) \in \mathbb{Z}[x, y]$ be an irreducible polynomial of degree $\delta$. Suppose $X$ and $Y$ are upper bounds for the desired solution $(x_0, y_0)$ and let $W = \max_{i,j} |f_{i,j}|X^i Y^i$. If $XY < W^{1/\delta}$, then we can find all solutions $(x_0, y_0)$ of $f(x, y) = 0$ bounded by $|x_0| \leq X$ and $|y_0| \leq Y$ in time polynomial in $(\log W, 2^\delta)$.

There is a generalisation of Coppersmith's method to multivariate polynomials. The basic idea follows that of the univariate case; we construct a similar lattice whose rows encode the coefficients of some *shift* polynomials which are multiples of powers of the modulus. After reducing the lattice basis, we obtain a set of multivariate polynomials which share the target roots over the integers. We can then use Gröbner basis techniques or compute resultants to solve the system of equations and recover the roots. As the polynomials we obtain after the lattice reduction step are not guaranteed to be algebraically independent (i.e. they may share a non-trivial factor), there is no guarantee that the system will be solvable. Therefore, this generalisation is heuristical only, though it tends to work quite well in practice. We also note that the bounds analysis typically depends heavily on the monomials that appear in the polynomial of interest as well as the shift polynomials used since the bounds depend on the determinant of the lattice. To calculate the upper bounds for the roots which we might expect the algorithm to successfully find, we combine the LLL bounds from Theorem 2.15 with the following result of Howgrave-Graham:

**Theorem 3.4 (Howgrave-Graham).** Let $h(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ be a polynomial consisting of $\omega$ monomials. If

(1) $f(r_1, \ldots, r_n) = 0 \pmod N$ for some $|r_1| < X_1, \ldots, |r_n| < X_n$
(2) $\|h(x_1 X_1, \ldots, x_n X_n)\| < \frac{N}{\sqrt{\omega}}$

Then $f(r_1, \ldots, r_n) = 0$ holds over the integers.

In practice, performing this analysis may be tedious and difficult, especially for lattices that aren't full-rank, so a more empirical approach is often used. For a more in-depth understanding of Coppersmith's method, see [Cop96, JM06, Cor07, May03].

## 3.2 Knapsack Problem

The knapsack problem is a well-known NP-complete computational problem that has been used as a trapdoor in some public key cryptosystems [MH78, CR88, Yas07]. The most common version of the knapsack problem in cryptography and cryptanalysis is the subset sum problem which involves finding a subset of a given set of numbers that sum to a given target. In this section, we will study a special case of this subset sum problem as well as the modular subset sum problem and further generalisations.

### 3.2.1 Low-density Subset Sum Problems

**Definition 3.5 (Subset Sum Problem).** Given positive integers $a_1, \ldots, a_n$ (the weights) and a target integer $s$, find some subset of the $a_i$ that sum to $s$. That is, find $e_1, \ldots, e_n$ with $e_i \in \{0, 1\}$ such that

$$\sum_{i=1}^{n} e_i a_i = s$$

Many cryptosystems based on the subset sum problem have been shown to be insecure [Odl91] through algorithms that solve special "low-density" subset sum problem instances. The *density* of a set of weights $a_1, \ldots, a_n$ is defined by

$$d = \frac{n}{\log_2 \max(a_i)}$$

[LO85] gives the LO algorithm which can solve subset sum problem instances with $d < 0.6463$ given access to a SVP oracle. Similarly, [Cos+92] gives the CJLOSS algorithm which is a slight modification of

the LO algorithm that improves the bound to $d < 0.9408$. Both of these algorithms are based on lattice reduction and have been shown to be quite practical [Cos+92].

The strategy is to construct a lattice which contains a vector encoding the $e_i$ as a short vector. A simple lattice which follows this idea is generated by the rows of the following basis matrix:

$$\mathbf{B} = \begin{bmatrix} 1 & & & & a_1 \\ & 1 & & & a_2 \\ & & \ddots & & \vdots \\ & & & 1 & a_n \\ & & & & s \end{bmatrix}$$

Notice that the linear combination $\mathbf{t} = (e_1, \ldots, e_n, -1)$ generates the (short) vector $\mathbf{x}_1 = (e_1, \ldots, e_n, 0)$. That is, $\mathbf{tB} = \mathbf{x}_1$. So we might intuitively expect lattice reduction to help us find this vector. The CJLOSS algorithm uses the slightly different lattice with basis $\mathbf{B}'$ given by

$$\mathbf{B}' = \begin{bmatrix} 1 & & & & Na_1 \\ & 1 & & & Na_2 \\ & & \ddots & & \vdots \\ & & & 1 & Na_n \\ \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} & Ns \end{bmatrix}$$

where $N > \sqrt{n}$ is an integer. In this case, the same linear combination $\mathbf{t} = (e_1, \ldots, e_n, -1)$ generates the vector $\mathbf{x}_2 = (e_1 - \frac{1}{2}, \ldots, e_n - \frac{1}{2}, 0)$, and so we always have $\|\mathbf{x}_2\| = \frac{1}{2}\sqrt{n}$. In [Cos+92] it is shown that the probability, $P$, of $\mathbf{x}_2$ not being the unique shortest vector in the lattice is bounded by

$$P \leq n(4n\sqrt{n} + 1)\frac{2^{c_0 n}}{\max(a_i)}$$

where $c_0 = 1.0628\ldots$. This upper bound tends towards 0 as $\max(a_i) > 2^{c_0 n}$, and so we get

$$\begin{aligned} & \max(a_i) > 2^{c_0 n} \\ \implies & \log_2 \max(a_i) > c_0 n \\ \implies & \frac{1}{c_0} > \frac{n}{\log_2 \max(a_i)} \\ \implies & d < \frac{1}{c_0} \\ \implies & d < 0.9408\ldots \end{aligned}$$

Therefore, almost all subset sum problems with density $d < 0.9408$ can be efficiently solved given a SVP oracle. It is important to note that this result is theoretical and proven with the hypothetical existence of a SVP oracle. In reality, although such an oracle does not exist, LLL often finds a sufficiently short vector that allows us to solve the subset sum problem. We also note that the upper bound on the density is theoretical as well and in practice it may even be possible to solve higher density subset sum problems with this approach.

**Example 3.6.** Let $(a_1, a_2, a_3, a_4, a_5, a_6) = (83, 59, 47, 81, 76, 51)$ be the $n = 6$ weights of a subset sum problem and let $s = 291$ be the target. The density of this subset sum problem is $d = n/\log_2 \max(a_i) = 6/6.375 = 0.9412$. Although the density is slightly higher than the theoretical upper bound, we will use the CJLOSS algorithm and show that it can solve this subset sum problem. With $N = 3$ we construct

the lattice generated by the rows of $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} 1 & & & & Na_1 \\ & 1 & & & Na_2 \\ & & \ddots & & \vdots \\ & & & 1 & Na_n \\ \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} & Ns \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 3 \cdot 83 \\ 0 & 1 & 0 & 0 & 0 & 0 & 3 \cdot 59 \\ 0 & 0 & 1 & 0 & 0 & 0 & 3 \cdot 47 \\ 0 & 0 & 0 & 1 & 0 & 0 & 3 \cdot 81 \\ 0 & 0 & 0 & 0 & 1 & 0 & 3 \cdot 76 \\ 0 & 0 & 0 & 0 & 0 & 1 & 3 \cdot 51 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 3 \cdot 291 \end{bmatrix}$$

Running LLL on this basis yields the reduced basis $\mathbf{B}'$:

$$\mathbf{B}' = \frac{1}{2} \begin{bmatrix} -1 & 1 & 1 & -1 & -1 & -1 & 0 \\ 1 & 1 & 1 & -1 & -1 & 3 & 0 \\ 2 & -2 & 2 & 2 & -4 & 0 & 0 \\ 1 & 3 & -3 & 3 & -3 & 1 & 0 \\ 2 & -2 & -2 & -4 & 0 & 0 & 0 \\ -3 & -1 & -3 & -1 & -1 & 1 & 0 \\ 0 & -2 & 0 & 0 & -2 & -2 & -6 \end{bmatrix}$$

The first row $\mathbf{b}_1 = (b_1, \ldots, b_7) = (-\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, 0)$ is our vector $\mathbf{x} = (e_1 - \frac{1}{2}, \ldots, e_6 - \frac{1}{2}, 0)$ (or $-\mathbf{x}$) which encodes the $e_i$. Thus, we recover the $e_i$ by computing $(b_1 + \frac{1}{2}, \ldots, b_6 + \frac{1}{2})$ (or $(\frac{1}{2} - b_1, \ldots \frac{1}{2} - b_6)$). In this case, doing the latter gives us

$$(e_1, e_2, e_3, e_4, e_5, e_6) = (1, 0, 0, 1, 1, 1)$$

which indeed satisfies $\sum_{i=1}^{n} e_i a_i = s$.

### 3.2.2 Low-density Subset Sum Problems: Extensions and Generalisations

It turns out that the approach of the previous section can be extended to the multiple subset sum problem, the modular subset sum problem and the multiple modular subset sum problem. We have the following definitions for these problems.

**Definition 3.7 (Multiple Subset Sum Problem).** Given positive integers $a_{1,1}, \ldots, a_{k,n}$ (the weights) and target integers $s_1, \ldots, s_k$, find $e_1, \ldots, e_n$ with $e_i \in \{0, 1\}$ such that

$$\sum_{i=1}^{n} e_i a_{j,i} = s_j$$

for all $1 \leq j \leq k$.

**Definition 3.8 (Modular Subset Sum Problem).** Given positive integers $a_1, \ldots, a_n$ (the weights), a target integer $s$, and a modulus $M$, find $e_1, \ldots, e_n$ with $e_i \in \{0, 1\}$ such that

$$\sum_{i=1}^{n} e_i a_i = s \pmod{M}$$

**Definition 3.9 (Multiple Modular Subset Sum Problem).** Given positive integers $a_{1,1}, \ldots, a_{k,n}$ (the weights), target integers $s_1, \ldots, s_k$, and a modulus $M$, find $e_1, \ldots, e_n$ with $e_i \in \{0, 1\}$ such that

$$\sum_{i=1}^{n} e_i a_{j,i} = s_j \pmod{M}$$

for all $1 \leq j \leq k$.

The density of the multiple subset sum problem is defined as $d = \frac{n}{k \cdot \log_2 \max(a_{j,i})}$, while the density of the multiple modular subset sum problem is defined as $d = \frac{n}{k \cdot \log_2 M}$. We also note that the (modular) subset sum problem is simply the multiple (modular) subset sum problem with $k = 1$.

In [PZ16], it is shown that the multiple subset sum problem can be solved with the same density bound of $d < 0.9408$ using the lattice with basis $\mathbf{B}$ given by:

$$\mathbf{B} = \begin{bmatrix} 1 & & & & 0 & Na_{1,1} & Na_{2,1} & \cdots & Na_{k,1} \\ & 1 & & & 0 & Na_{1,2} & Na_{2,2} & \cdots & Na_{k,2} \\ & & \ddots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ & & & 1 & 0 & Na_{1,n} & Na_{2,n} & \cdots & Na_{k,n} \\ \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} & \frac{1}{2} & Ns_1 & Ns_2 & \cdots & Ns_k \end{bmatrix}$$

with $N > \sqrt{\frac{n+1}{4}}$. Similar to before, the linear combination $(e_1, \ldots, e_n, -1)$ generates the short vector $\mathbf{x} = (e_1 - \frac{1}{2}, \ldots, e_n - \frac{1}{2}, -\frac{1}{2}, 0, \ldots, 0)$ and so we expect lattice reduction to reveal this target vector.

[PZ16] also shows that the modular multiple subset sum problem can be solved when $d < 0.9408$ and $k = o\left(\frac{n}{\log_2((n+1)\sqrt{n}+1)}\right)$ using the lattice with basis $\mathbf{B}'$ given by:

$$\mathbf{B}' = \begin{bmatrix} 1 & & & & 0 & Na_{1,1} & Na_{2,1} & \cdots & Na_{k,1} \\ & 1 & & & 0 & Na_{1,2} & Na_{2,2} & \cdots & Na_{k,2} \\ & & \ddots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ & & & 1 & 0 & Na_{1,n} & Na_{2,n} & \cdots & Na_{k,n} \\ & & & & & NM & & & \\ & & & & & & NM & & \\ & & & & & & & \ddots & \\ & & & & & & & & NM \\ \frac{1}{2} & \frac{1}{2} & \cdots & \frac{1}{2} & \frac{1}{2} & Ns_1 & Ns_2 & \cdots & Ns_k \end{bmatrix}$$

To see why the target vector is in this lattice, we rewrite the modular equations $\sum_{i=1}^{n} e_i a_{j,i} = s_j \pmod{M}$ as $\sum_{i=1}^{n} e_i a_{j,i} = s_j + \ell_j M$ and notice that the linear combination $(e_1, \ldots, e_n, -\ell_1, \ldots, -\ell_k, -1)$ generates the target vector $\mathbf{x} = (e_1 - \frac{1}{2}, \ldots, e_n - \frac{1}{2}, -\frac{1}{2}, 0, \ldots, 0)$.

## 3.3 Hidden Number Problem

The hidden number problem (HNP) was introduced in [BV96] for the purpose of proving results about the bit security of the Diffie-Hellman key-exchange protocol. At a high level, the HNP deals with recovering a secret "hidden" number given some partial knowledge of its linear relations, so it has naturally found further usefulness in cryptanalysis and especially side-channel attacks. In this section, we will study the hidden number problem as well as the extended hidden number problem as formulated in [HR07].

### 3.3.1 Hidden Number Problem: An Overview

The original formulation of the HNP in [BV96] is in terms of finding a secret integer $\alpha$ modulo a public prime $p$ when given the most significant bits of a number of $t_i\alpha \mod p$, where the $t_i$ are random and known. We follow the reformulation given in [BH19] which is a slight variant that models the problem as seeking a solution to a system of linear equations.

**Definition 3.10 (Hidden Number Problem).** Let $p$ be a prime and let $\alpha \in [1, p-1]$ be a secret integer. Recover $\alpha$ given $m$ pairs of integers $\{(t_i, a_i)\}_{i=1}^{m}$ such that

$$\beta_i - t_i\alpha + a_i = 0 \pmod{p}$$

where the $\beta_i$ are unknown and satisfy $|\beta_i| < B$ for some $B < p$.

For appropriate parameters, the HNP can be solved via a reduction to the closest vector problem. Consider the matrix with basis $\mathbf{B}$ given by

$$\mathbf{B} = \begin{bmatrix} p & & & & \\ & p & & & \\ & & \ddots & & \\ & & & p & \\ t_1 & t_2 & \cdots & t_m & 1/p \end{bmatrix}$$

By rewriting the HNP equations as $\beta_i + a_i = t_i\alpha + k_i p$ for integers $k_i$, we see that the linear combination $\mathbf{x} = (k_1, \ldots, k_m, \alpha)$ generates the lattice vector $\mathbf{xB} = (\beta_1 + a_1, \ldots, \beta_m + a_m, \alpha/p)$. Defining $\mathbf{t} = (a_1, \ldots, a_m, 0)$ and $\mathbf{u} = (\beta_1, \ldots, \beta_m, \alpha/p)$, we notice that $\mathbf{xB} - \mathbf{t} = \mathbf{u}$ where the length of $\mathbf{u}$ is bounded above by $\sqrt{m+1}B$, whereas the lattice determinant is $p^{m-1}$. Therefore, we can reasonably expect an approximate CVP algorithm to reveal the vector $\mathbf{u}$ from which we can read off the secret integer $\alpha$ by multiplying the last entry by $p$.

In [BV96], the authors prove that this approach is successful using Babai's algorithm with LLL when $m = 2\lceil\sqrt{\log p}\rceil$ and $B \leq p/2^k$ where $k = \lceil\sqrt{\log p}\rceil + \lceil\log\log p\rceil$. In practice, the HNP can be solved with looser bounds on the parameters, especially in smaller dimensions and with optimisations such as recentering [BH19]. It has also been shown that an SVP approach using Kannan's embedding method is often more effective than the CVP approach [Ben+14, Sun+21].

With the SVP approach, we embed the CVP target vector as a row in the lattice basis to get the basis $\mathbf{B}'$:

$$\mathbf{B}' = \begin{bmatrix} p & & & & & \\ & p & & & & \\ & & \ddots & & & \\ & & & p & & \\ t_1 & t_2 & \cdots & t_m & B/p & \\ a_1 & a_2 & \cdots & a_m & & B \end{bmatrix}$$

This lattice contains the vector

$$\mathbf{u}' = (\beta_1, \ldots, \beta_m, \alpha B/p, -B)$$

generated by the linear combination $(k_1, \ldots, k_m, \alpha, -1)$. We have $\|\mathbf{u}'\| < \sqrt{m+2}B$ and so Proposition 2.15 suggests that we are likely to find this short vector among the basis vectors of an LLL-reduced basis. Notably, the shorter vector $(0, \ldots, 0, B, 0)$ is in this lattice, generated by the linear combination $(-t_1, \ldots, -t_m, p, 0)$, so $\mathbf{u}'$ is more likely to be the second vector of an LLL-reduced basis.

**Example 3.11.** Let $p = 401$ and $(t_1, t_2, t_3) = (143, 293, 304)$. We will use the SVP approach to solving the HNP to recover the secret integer $\alpha = 309$. Suppose we are given that $\beta_i - t_i\alpha + a_i = 0 \pmod{p}$ for $i \in \{1, 2, 3\}$ where $(a_1, a_2, a_3) = (62, 300, 86)$ and furthermore, that $\beta_i < B = 20$. We construct the matrix generated by the rows of $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} p & 0 & 0 & 0 & 0 \\ 0 & p & 0 & 0 & 0 \\ 0 & 0 & p & 0 & 0 \\ t_1 & t_2 & t_3 & B/p & 0 \\ a_1 & a_2 & a_3 & 0 & B \end{bmatrix} = \begin{bmatrix} 401 & 0 & 0 & 0 & 0 \\ 0 & 401 & 0 & 0 & 0 \\ 0 & 0 & 401 & 0 & 0 \\ 143 & 293 & 304 & \frac{20}{401} & 0 \\ 62 & 300 & 86 & 0 & 20 \end{bmatrix}$$

Running LLL on this basis gives us the LLL-reduced basis $\mathbf{B}'$:

$$\mathbf{B}' = \begin{bmatrix} 0 & 0 & 0 & 20 & 0 \\ -15 & -12 & -16 & \frac{1840}{401} & 20 \\ 24 & -5 & -6 & -\frac{1800}{401} & 20 \\ 6 & -42 & -5 & -\frac{1440}{401} & -40 \\ -11 & -1 & 57 & -\frac{3880}{401} & 20 \end{bmatrix}$$

As expected, the first basis vector is $(0, 0, 0, B, 0)$. To recover $\alpha$, we look through the other basis vectors. We note that our target lattice vector $\mathbf{u} = (\beta_1, \beta_2, \beta_3, \alpha B/p, -B)$ is not among the basis vectors as no basis vector has $-20$ as their last element. However, $-\mathbf{u}$ has the same length as $\mathbf{u}$, and is potentially among the basis vectors. Indeed, $-\mathbf{u}$ is found in the second basis vector and we also find that this yields $\beta_i < B$ for $i \in \{1, 2, 3\}$. Therefore, we read off the secret integer $\alpha$ from the second last entry in $-\mathbf{u}$ to find that $\alpha = -(\frac{1840}{401}) \cdot \frac{401}{20} \mod p = \alpha = 309$.

### 3.3.2 Extended Hidden Number Problem

The extended hidden number problem formulated in [HR07] extends the HNP to the case in which there are multiple chunks of information known about linear relations of the secret integer. Additionally, it simultaneously deals with the case in which multiple chunks of the secret integer are known.

**Definition 3.12 (Extended Hidden Number Problem [HR07]).** Let $p$ be a prime and let $x \in [1, p-1]$ be a secret integer such that

$$x = \bar{x} + \sum_{j=1}^{m} 2^{\pi_j} x_j$$

where the integers $\bar{x}$ and $\pi_j$ are known, and the unknown integers $x_j$ satisfy $0 \leq x_j < 2^{\nu_j}$ for known integers $\nu_j$. Suppose we are given $d$ equations

$$\alpha_i \sum_{j=1}^{m} 2^{\pi_j} x_j + \sum_{j=1}^{l_i} \rho_{i,j} k_{i,j} = \beta_i - \alpha_i \bar{x} \pmod{p}$$

for $1 \leq i \leq d$ where $\alpha_i \neq 0 \pmod{p}$, $\rho_{i,j}$ and $\beta_i$ are known integers. The unknown integers $k_{i,j}$ are bounded by $0 \leq k_{i,j} < 2^{\mu_{i,j}}$ where the $\mu_{i,j}$ are known. The extended hidden number problem (EHNP) is to find $x$. The EHNP instance is represented by

$$\left(\bar{x}, p, \{\pi_j, \nu_j\}_{j=1}^{m}, \left\{\alpha_i, \{\rho_{i,j}, \mu_{i,j}\}_{j=1}^{l_i}, \beta_i\right\}_{i=1}^{d}\right)$$

As with the hidden number problem, we model the situation as a CVP instance. The main idea behind the lattice basis used to solve the EHNP is similar to that of the regular HNP except the EHNP lattice involves factors to deal with the varying sizes of the unknown chunks. For a $\delta > 0$ (which will be chosen later), we construct the EHNP lattice basis $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} p \cdot \mathbf{I}_d & & \\ \mathbf{A} & \mathbf{X} & \\ \mathbf{R} & & \mathbf{K} \end{bmatrix}$$

with the following definitions:

$$\mathbf{A} = \begin{bmatrix} \alpha_1 2^{\pi_1} & \alpha_2 2^{\pi_1} & \cdots & \alpha_d 2^{\pi_1} \\ \alpha_1 2^{\pi_2} & \alpha_2 2^{\pi_2} & \cdots & \alpha_d 2^{\pi_2} \\ \vdots & \ddots & & \vdots \\ \alpha_1 2^{\pi_m} & \alpha_2 2^{\pi_m} & \cdots & \alpha_d 2^{\pi_m} \end{bmatrix} \qquad \mathbf{X} = \operatorname{diag}\left(\frac{\delta}{2^{\nu_1}}, \frac{\delta}{2^{\nu_2}}, \ldots, \frac{\delta}{2^{\nu_m}}\right)$$

$$\mathbf{R} = \begin{bmatrix} \rho_{1,1} & & & \\ \vdots & & & \\ \rho_{1,l_1} & & & \\ & \ddots & & \\ & & \rho_{d,1} & \\ & & \vdots & \\ & & \rho_{d,l_d} & \end{bmatrix} \qquad \mathbf{K} = \operatorname{diag}\left(\frac{\delta}{2^{\mu_{1,1}}}, \ldots, \frac{\delta}{2^{\mu_{1,l_1}}}, \ldots, \frac{\delta}{2^{\mu_{d,1}}}, \ldots, \frac{\delta}{2^{\mu_{d,l_d}}}\right)$$

To understand what vector we should target with CVP, we rewrite the EHNP equations as

$$\alpha_i \sum_{j=1}^{m} 2^{\pi_j} x_j + \sum_{j=1}^{l_i} \rho_{i,j} k_{i,j} + r_i p = \beta_i - \alpha_i \bar{x}$$

for integers $r_i$. Now, consider the lattice vector $\mathbf{u}$ generated by the linear combination $\mathbf{x}$ which contains secret information:

$$\mathbf{x} = (r_1, \ldots, r_d, x_1, \ldots, x_m, k_{1,1}, \ldots, k_{1,l_1}, \ldots, k_{d,1}, \ldots, k_{d,l_d})$$

We have

$$\mathbf{xB} = \mathbf{u} = \left(\beta_1 - \alpha_1 \bar{x}, \ldots, \beta_d - \alpha_d \bar{x}, \frac{x_1 \delta}{2^{\nu_1}}, \ldots, \frac{x_m \delta}{2^{\nu_m}}, \frac{k_{1,1}\delta}{2^{\mu_{1,1}}}, \ldots, \frac{k_{1,l_1}\delta}{2^{\mu_{1,l_1}}}, \ldots, \frac{k_{d,1}\delta}{2^{\mu_{d,1}}}, \ldots, \frac{k_{d,l_d}\delta}{2^{\mu_{d,l_d}}}\right)$$

Then, letting

$$\mathbf{w} = \left(\beta_1 - \alpha_1 \bar{x}, \ldots, \beta_d - \alpha_d \bar{x}, \frac{\delta}{2}, \ldots, \frac{\delta}{2}, \frac{\delta}{2}, \ldots, \frac{\delta}{2}, \ldots, \frac{\delta}{2}, \ldots, \frac{\delta}{2}\right)$$

we notice that $\mathbf{w}$ is close to the lattice vector $\mathbf{u}$. Therefore, by solving the CVP instance with $\mathbf{w}$ as the target vector, we may reveal the lattice vector $\mathbf{u}$ that encodes the secret chunks $x_j$ in the $(d+1)$st to $(d+m)$th entries.

It remains to choose an appropriate $\delta$. The proof of correctness of this algorithm in [HR07] requires $\delta$ to be chosen such that $0 < \kappa_D \delta < 1$ where

$$L = \sum_{i=1}^{d} l_i, \qquad D = d + m + L, \qquad \kappa_D = \frac{2^{D/4}(m+L)^{1/2} + 1}{2}$$

When this is the case, the approach described above to solving the EHNP succeeds with probability

$$P > 1 - \frac{(2\kappa_D)^{L+m} 2^{\tau+\xi}}{p^d}$$

where $\tau = \sum_{j=1}^{m} \nu_j$ and $\xi = \sum_{i=1}^{d} \sum_{j=1}^{l_i} \mu_{i,j}$. As expected, we have more success when we have more equations (given by the parameter $d$), and less success when the number of chunks and amount of unknown information is more (given by the parameters $m, \nu_j, l_i, \mu_{i,j}$).

# 4 Lattice Attacks

In this section, we study some attacks on cryptosystems made possible by lattice reduction techniques. We will focus on transforming cryptographic problems into lattice-based problems which we have already learnt how to solve.

## 4.1 RSA Stereotyped Message

One of the first and most well-known applications of Coppersmith's work on finding small roots of modular polynomials is an attack on low-exponent RSA. This attack is due to Coppersmith [Cop97] and directly applies Coppersmith's method to recover the full plaintext of an RSA ciphertext when the public exponent is 3 and at least two thirds of the plaintext is known.

### 4.1.1 Simple Case

Let $N$ be an RSA modulus, $e$ a (small) public exponent, and $c = m^e \pmod N$ a given ciphertext for the message $m$. Suppose that $m$ is of the form $m = m' + x_0$ for known $m'$. If $x_0$ is small enough, we can formulate the RSA equation as a small roots problem and try to solve it with Coppersmith's method. We have

$$\begin{aligned} & c = m^e \pmod N \\ \implies & c = (m' + x_0)^e \pmod N \\ \implies & (m' + x_0)^e - c = 0 \pmod N \end{aligned}$$

Writing $f(x) = (m' + x)^e - c \pmod N$, we have a modular polynomial of degree $e$ in $x$. From Theorem 3.2, we can solve for $x$ to recover the root $x_0$ using Coppersmith's method when $|x_0| \leq \frac{1}{2} N^{\frac{1}{e}}$. In the case where $e = 3$, this turns out to be quite practical and implies that the unknown part of the plaintext can be recovered if we know two thirds of the rest.

**Example 4.1.** As a motivating example, consider the RSA modulus

$$N = 31811053356484684632768156296980630626705075736074 1$$

with $e = 3$ and the *stereotyped* message $m =$ "`my secret pin is XXXX`" where `XXXX` is an unknown four digit pin code. We are given the ciphertext

$$c = m^e = 312332738778608882264230787188876936416561274050341 \pmod N$$

Because we know a part of the plaintext, we can write $m$ as $m = m' + x_0$ where $m'$ is the known part, and $x_0$ is the unknown part. For this expression to make sense, we convert the known message bytes "`my secret pin is \x00\x00\x00\x00`" to an integer to get

$$m' = 159995190028598044409165991369948950987562188537856$$

Since the pin is 4 digits, we may assume that $x_0$ is smaller than $(\text{FFFFFFFF})_{16}$. Therefore, we have the modular polynomial in $x$

$$f(x) = (m' + x)^e - c$$

which has a root $|x_0| < 2^{32}$. This root is small compared to the size of $N$, so we can recover it by Coppersmith's theorem. Performing Coppersmith's method on this polynomial recovers the small root $x_0 = 825439031$ whose bytes representation reveals the secret pin `1337`.

### 4.1.2 Many Unknown Parts

With an extension of Coppersmith's method to many variables, this stereotyped message attack can be adapted to work in the more general situation where (non-contiguous) unknown parts of the plaintext are scattered amongst a larger portion of known plaintext and small public exponent is used. For example, consider the message

$$m = \text{"`my four letter username is XXXX and my secret four digit pin code is YYYY`"}$$

where the `XXXX` and `YYYY` are unknown parts of the message we wish to recover. We can rewrite $m$ as $m = m' + 2^{t_x} x_0 + 2^{t_y} y_0$ where $x_0$ and $y_0$ represent the unknown parts of the message and $m'$ represents the known part:

$$m' = \text{“}\texttt{my four letter username is \x00\x00\x00\x00}$$
$$\texttt{and my secret four digit pin code is \x00\x00\x00\x00}\text{”}$$

The $2^{t_x}$ and $2^{t_y}$ factors exist to capture position of the unknown parts in the message. In this example, $t_x = 42 \times 8 = 336$ and since $t_y$ starts at the least significant bit, $t_y = 0$. To recover $x_0$ and $y_0$, we construct the modular polynomial $f(x, y)$ of degree $e$ in $x$ and $y$:

$$f(x, y) = (m' + 2_x^t x + 2_y^t y)^e - c$$

and use Coppersmith's method for multivariate polynomials to recover the small root $(x_0, y_0)$.

## 4.2 Partial Key Exposure Attacks on RSA

In 1990, Wiener gave an attack on the RSA cryptosystem showing that it can be broken if the private exponent $d$ is smaller than $N^{0.25}$ [Wie90]. A decade later, the bound on $d$ was improved by Boneh and Durfee to $d < N^{0.292}$ with a new approach using lattice reduction techniques [BD99]. Although there have been no new results improving this bound since, the Boneh-Durfee attack has been revisited many times and it has been shown that the bound can be improved given the relaxed condition of partial knowledge of the private exponent [BM03, Ern+05].

### 4.2.1 Boneh-Durfee Attack

Let $N = pq$ be a (balanced) RSA modulus, $e$ a public exponent, and $d$ the corresponding private exponent where $ed = 1 \pmod{\varphi(N)}$ and $\varphi(N) = N - p - q + 1$. Suppose that $d < N^\delta$ with $\delta < 0.292$. From the RSA key equation, we have

$$\begin{aligned} ed &= 1 \pmod{\varphi(N)} \\ \implies ed &= 1 + k(N - p - q + 1) \\ \implies 1 + k(N - p - q + 1) &= 0 \pmod{e} \\ \implies 1 + 2k\left(\frac{N+1}{2} - \frac{p+q}{2}\right) &= 0 \pmod{e} \end{aligned}$$

where $k = \frac{ed-1}{\varphi(N)}$, and the last line follows since $N+1$ and $p+q$ are necessarily always even. For simplicity, we assume that $e < \varphi(N)$, and so we get

$$\begin{aligned} k = \frac{ed-1}{\varphi(N)} &< d \\ \implies k &< N^\delta \end{aligned}$$

Furthermore, since the modulus is balanced, then $\frac{p+q}{2} < N^{0.5}$. Thus, we consider the bivariate modular polynomial

$$f(x, y) = 1 + 2x\left(\frac{N+1}{2} - y\right) \pmod{e}$$

which has the "small" root $(x, y) = (k, (p+q)/2)$. In [BD99], the authors use a Coppersmith approach with specific shift polynomials to show that the root can be recovered as long as $\delta < 1 - 1/\sqrt{2} \approx 0.292$. A more generic approach to finding small roots such as the strategy described in [JM06] may be used to recover the roots, although with less guarantees on the bounds.

After finding the root $(k, s)$, it is easy to recover the private key $d$ by evaluating $f(k, s)$ over the integers to get $ed = f(k, s)$ and then dividing the result by $e$.

### 4.2.2 Partial Key Exposure Attack

We now give an overview of the partial key exposure attack first introduced in [Ern+05], following the simplification of [SGM10] whose general approach is quite similar to that of the Boneh-Durfee attack.

In this setting, we have a balanced RSA modulus $N = pq$, a public exponent $e$, and the corresponding private exponent $d$. Furthermore, we are given that $d < N^\delta$ as well as $(\delta - \gamma)\log_2 N$ MSBs of $d$. That is, we know an integer $d_0$ such that $|d - d_0| < N^\gamma$. If $\delta < 0.292$, then we can directly apply the Boneh-Durfee attack without using the additional partial knowledge of $d$. Although the Boneh-Durfee attack cannot be used for $\delta > 0.292$, it is still helpful to revisit the reformulation of the RSA key equation used in the attack:

$$1 + 2k\left(\frac{N+1}{2} - \frac{p+q}{2}\right) = 0 \pmod{e}$$

They key idea is that we can use the known bits of $d$ to approximate $k$ so that the bounds on the roots that we need to solve for are lowered. Let $d = d_0 + d_1$ where $d_1$ is the remainder of the unknown bits of $d$. Note that

$$\begin{aligned} k &= \frac{ed - 1}{\varphi(N)} \\ &= \frac{e(d_0 + d_1) - 1}{\varphi(N)} \\ &\approx \frac{ed_0}{N} \end{aligned}$$

So $k_0 = \lfloor \frac{ed_0}{N} \rfloor$ is a good approximation for $k$. In [BM03] it is shown that this approximation satisfies $|k - k_0| < 4N^\lambda$ with $\lambda = \max\{\gamma, \delta - \frac{1}{2}\}$. Therefore, writing $k = k_0 + k_1$ where $k_1$ is the remainder of the unknown bits of $k$, we find that the polynomial

$$f(x, y) = 1 + 2(k_0 + x)\left(\frac{N+1}{2} - y\right) \pmod{e}$$

has the "small" root $(x, y) = (k_1, (p+q)/2)$ for appropriate $\delta$ and $\gamma$.

## 4.3 ECDSA with Bad Nonces

### 4.3.1 ECDSA with $k = z \oplus d$

In the ECDSA signature scheme, the generation of the nonce must be handled with a lot of care. It must not only be secret, but also needs to be sufficiently random and reveal no additional relations or information about the private key. In this section, we study an interesting situation in which the nonce is chosen to be the result of XORing the private key with the hash of the message being signed. Since the private key is unknown, such a nonce is unpredictable. However, we will see that using this nonce generation reveals linear relations in the bits of the private key which can then be transformed into a lattice-based problem instance. The approach we present comes from the challenge *Signature* from the *TSJ CTF 2022* Capture-The-Flag event [map22].

Suppose we are given $\ell$ ECDSA signatures $(z_i, r_i, s_i)$. Each signature is calculated using the nonce $k_i = z_i \oplus d$ where $d$ is the private signing key. $z_i$ is the hash of each message and the $r_i$ and $s_i$ satisfy the ECDSA equation

$$s_i = k_i^{-1}(z_i + r_i d) \pmod{n}$$

where $n$ is the order of the elliptic curve. Rearranging, we can write this equation more cleanly as

$$s_i k_i = z_i + r_i d \pmod{n}$$

Note that the only unknowns in this equation are $k_i$ and $d$. We know that these are related however, specifically

$$\begin{aligned} k_i &= z_i \oplus d \\ &= z_i + d - \sum_{j=1}^{m} 2^j z_{i,[j]} d_{[j]} \end{aligned}$$

where $x_{[j]}$ denotes the $j$th (least significant) bit of $x$ and $m$ is the number of bits in $z_i$ and $d$. Substituting this expression for $k_i$ into the ECDSA equation, we have

$$s_i(z_i + d - \sum_{j=1}^{m} 2^j z_{i,[j]} d_{[j]}) = z_i + r_i d \pmod{n}$$

Furthermore, if we write $d$ in terms of its bits, we see that we have a linear equation where the only unknowns are the $d_{[j]}$:

$$s_i(z_i + \sum_{j=1}^{m} 2^{j-1} d_{[j]} - \sum_{j=1}^{m} 2^j z_{i,[j]} d_{[j]}) = z_i + r_i \sum_{j=1}^{m} 2^{j-1} d_{[j]} \pmod{n}$$

For simplicity, we can write this as

$$\sum_{j=1}^{m} a_{i,j} d_{[j]} = b_i \pmod{n}$$

where we understand $a_{i,j}$ and $b_i$ to be values that we can efficiently compute from $z_i, r_i$ and $s_i$.

Noting that we have $\ell$ such relations, this resembles a multiple modular subset sum problem instance. Since the bit size of the weights is the same as the bit size of $d$, the problem becomes solvable as soon as $\ell \geq 2$, i.e. when the density is $\leq 0.5$.

### 4.3.2 ECDSA with Biased Nonces

As we have seen in the previous section, the proper generation of the nonce when using the ECDSA signature scheme is crucial to its security. It turns out that even the slightest bias of less than one bit in the nonce generation can lead to a full recovery of the private key [Ara+20]. The attack of [Ara+20] uses a transformation into a hidden number problem instance and solves it using a Fourier analysis approach. In contrast, the lattice reduction approach to solving the underlying HNP instance as described in [HS01] will be more successful when a smaller number of signatures with larger biases are known. Such situations are of practical interest and have been seen in real world implementations [Ben+14, BH19].

Suppose we are given $\ell$ ECDSA signatures $(z_i, r_i, s_i)$. $z_i$ is the hash of each message and the $r_i$ and $s_i$ satisfy the ECDSA equation

$$\begin{aligned} s_i &= k_i^{-1}(z_i + r_i d) \pmod{n} \\ \implies s_i k_i &= z_i + r_i d \pmod{n} \end{aligned}$$

where $d$ is the private signing key, $n$ is the order of the elliptic curve and $k_i$ is the biased nonce. In the remainder of this section, we will study this attack with different interpretations of the word "biased". In each case, we perform some simple algebraic manipulation and transform the situation into a hidden number problem instance.

**Zero MSB** In this case, we assume that the nonces are generated such that the top $l$ bits of each $k_i$ are zero. Therefore, we have $|k_i| < 2^{\log_2 n - l}$, and so

$$\begin{aligned} s_i k_i &= z_i + r_i d \pmod{n} \\ \implies k_i - (s_i^{-1} r_i) d + (-s_i^{-1} z_i) &= 0 \pmod{n} \end{aligned}$$

is precisely a hidden number problem instance which can be solved given large enough $l$ and $\ell$.

**Zero LSB** If instead the $l$ LSBs of each $k_i$ are zero, then we can write $k_i = 2^l k_i'$ where $|k_i'| < 2^{\log_2 n - l}$. Then, we have

$$\begin{aligned} s_i k_i &= z_i + r_i d \pmod{n} \\ \implies s_i(2^l k_i') &= z_i + r_i d \pmod{n} \\ \implies k_i' - (2^{-l} s_i^{-1} r_i) d + (-2^{-l} s_i^{-1} z_i) &= 0 \pmod{n} \end{aligned}$$

which is a hidden number problem instance.

**Known MSB** Suppose that we know the $l$ MSBs of each $k_i$. Then we can write $k_i = 2^{\log_2 n - l} t_i + k_i'$ where $t_i$ is the known MSBs of $k_i$, and $|k_i'| < 2^{\log_2 n - l}$. Then, we have

$$\begin{aligned} s_i k_i &= z_i + r_i d \pmod n \\ \implies s_i(2^{\log_2 n - l} t_i + k_i') &= z_i + r_i d \pmod n \\ \implies k_i' - (s_i^{-1} r_i) d + (2^{\log_2 n - l} t_i - s_i^{-1} z_i) &= 0 \pmod n \end{aligned}$$

which is again a hidden number problem instance.

**Shared MSB** In this case, we are given that the $l$ MSBs of each $k_i$ are the same, but not necessarily what this shared value is. We write $k_i = 2^{\log_2 n - l} t + k_i'$ where $t$ is the unknown shared MSBs and $|k_i'| < 2^{\log_2 n - l}$. Substituting this expression for $k_i$ into the ECDSA equation, we have

$$\begin{aligned} s_i(2^{\log_2 n - l} t + k_i') &= z_i + r_i d \pmod n \\ \implies 2^{\log_2 n - l} t + k_i' - s_i^{-1} r_i d - s_i^{-1} z_i &= 0 \pmod n \end{aligned}$$

Therefore, to eliminate the $2^{\log_2 n - l} t$ term, we take the following $\ell - 1$ relations for $2 \le i \le \ell$ to be the equations in our hidden number problem instance:

$$\begin{aligned} (2^{\log_2 n - l} t + k_i' - s_i^{-1} r_i d - s_i^{-1} z_i) - (2^{\log_2 n - l} t + k_1' - s_1^{-1} r_1 d - s_1^{-1} z_1) &= 0 \pmod n \\ \implies (k_i' - k_1') - (s_i^{-1} r_i - s_1^{-1} r_1) d + (s_1^{-1} z_1 - s_i^{-1} z_i) &= 0 \pmod n \end{aligned}$$

### 4.3.3 ECDSA Key Disclosure Problem

The biased nonce situation of the previous section is generalised by the (EC)DSA key disclosure problem as described in [HR07]. In this setting, we are given (non-contiguous) chunks of both the nonces and the private signing key. Such a scenario may be realistic, for example, if an attacker is able to recover some bits of the nonces but not enough to apply the biased nonce attack and if some bits of the private key are known (e.g. from a side channel). We will see how this very naturally translates into an extended hidden number problem instance.

Suppose we are given $\ell$ ECDSA signatures $(z_i, r_i, s_i)$. $z_i$ is the hash of each message and the $r_i$ and $s_i$ satisfy the ECDSA equation

$$s_i = k_i^{-1}(z_i + r_i d) \pmod n$$

where $d$ is the private signing key, $n$ is the order of the elliptic curve and $k_i$ is the per-signature nonce. Furthermore, suppose that we know some (non-contiguous) chunks of each $k_i$ and $d$. So, we can write

$$d = \bar{d} + \sum_{j=1}^{m} 2^{\pi_j} d_j, \quad 0 \le d_j < 2^{\nu_j}$$

$$k_i = \bar{k_i} + \sum_{j=1}^{l_i} 2^{\lambda_{i,j}} k_{i,j}, \quad 0 \le k_{i,j} < 2^{\mu_{i,j}}$$

where we know all of $\bar{d}, \nu_j, \bar{k_i}, \pi_j, \lambda_{i,j}$ and $\mu_{i,j}$.

Substituting these expressions into the ECDSA equation, we get

$$\begin{aligned} s_i(\bar{k_i} + \sum_{j=1}^{l_i} 2^{\lambda_{i,j}} k_{i,j}) &= z_i + r_i(\bar{d} + \sum_{j=1}^{m} 2^{\pi_j} d_j) \pmod n \\ \implies r_i \sum_{j=1}^{m} 2^{\pi_j} d_j + \sum_{j=1}^{l_i} (-2^{\lambda_{i,j}} s_i) k_{i,j} &= (s_i \bar{k_i} - z_i) - r_i \bar{d} \pmod n \end{aligned}$$

which is the EHNP instance represented by

$$(\bar{d}, n, \{\pi_j, \nu_j\}_{j=1}^{m}, \left\{\alpha, \{\rho_{i,j}, \mu_{i,j}\}_{j=1}^{l_i}, \beta_i\right\}_{i=1}^{\ell})$$

where

$$\alpha = r_i, \qquad \rho_{i,j} = -2^{\lambda_{i,j}} s_i \pmod n, \qquad \beta_i = s_i \bar{k_i} - z_i \pmod n$$

## 4.4 Bleichenbacher's PKCS#1 v1.5 Padding Oracle Attack

As textbook RSA is well known to be susceptible to a variety of attacks, padding schemes are used in practice to preprocess the message before encryption. The PKCS#1 standard [KJR16] is one of the most widely implemented standards for RSA encryption and digital signatures. In 1998, Bleichenbacher presented an attack on version 1.5 [Kal98] of this standard which showed that an arbitrary ciphertext can be decrypted given access to a decryption oracle that reveals whether or not the corresponding plaintext has the correct format according to the padding scheme [Ble98]. Although the original attack description by Bleichenbacher was not in terms of lattices, there is a lattice formulation described by Nguyen [Ngu09]. This approach does not improve the query complexity of the attack (in fact, it is slightly worse), however, it may be more intuitive to understand and is an interesting application of lattice reduction. Furthermore, a lattice approach was used in an attack which improved Bleichenbacher's attack by using parallelisation, yielding practical results against modern TLS implementations [Ron+19]. In this section, we give a brief overview of the PKCS#1 v1.5 padding scheme and a lattice attack against it.

### 4.4.1 PKCS#1 v1.5

In this section, we describe the parts of the PKCS#1 v1.5 standard for encryption necessary to understand Bleichenbacher's attack.

| 00 | 02 | padding string | 00 | message $M$ |
|---|---|---|---|---|

Figure 12: PKCS#1 v1.5 block formatting for encryption.

Let $N$ be an RSA modulus and $e$ a public exponent. Let $k$ be the length of $N$ in bytes. In this scheme, $k$ must be at least 12. To encrypt a message $M$ of length $|M|$ bytes, a padding string $PS$ consisting of $k - 3 - |M|$ non-zero bytes is pseudorandomly generated. The padded encryption block $EB$ is then computed as $EB = 00||02||PS||00||M$ (where $||$ denotes concatenation) as per Figure 12. This block is converted to an integer $m$ which is then encrypted with RSA to get the ciphertext $c = m^e \pmod N$.

Given a ciphertext $c$, the message is recovered by decrypting $c$ with the private key $d$ to get $m = c^d \pmod N$. If the ciphertext is PKCS conforming, then $m$ can be converted to an encryption block starting with 0002 followed by a non-zero padding string that is separated from the message by a zero byte. By searching for the first zero byte starting from the padding string, the message is recovered by taking all bytes following the found zero byte.

### 4.4.2 The Attack (Lattice Version)

We now describe the lattice attack. The goal is to recover the message $m$ from a ciphertext $c = m^e \pmod N$. Assume that we have access to an oracle $\mathcal{O}_B$ which returns whether or not a given ciphertext is PKCS conforming. That is,

$$\mathcal{O}_B(c) = \begin{cases} 0 & \text{if } c \text{ is not PKCS conforming} \\ 1 & \text{if } c \text{ is PKCS conforming} \end{cases}$$

The existence of such an oracle is realistic, as many implementations often throw an error when an RSA ciphertext is not PKCS conforming. Importantly, we note that, if $\mathcal{O}_B(c) = 1$ for a given ciphertext, then the first two most significant bytes of $m$ are 00 and 02.

The key idea is to use the oracle to find many $r_i$ such that $\mathcal{O}_B(r_i^e c) = \mathcal{O}_B((r_i m)^e) = 1$. When choosing $r_i$ uniformly at random, we expect the oracle to return 1 with probability close to $1/256^2 = 1/65536$. Now suppose we have such integers $r_1, \ldots, r_\ell$. Since each $r_i m \mod N$ starts with 0002, we have

$$2B \leq r_i m \mod N < 3B$$

where $B = 2^{l-16}$ and $l$ is the length of $N$ in bits. Note that $B$ represents the bytes 0001 left shifted to

fill the size of $N$. Rearranging, we get

$$\begin{aligned} r_i m \mod N - 2B < B \\ \implies r_i m - 2B = k_i \pmod N \\ \implies k_i - r_i m + 2B = 0 \pmod N \end{aligned}$$

where $|k_i| < B$. This is precisely a hidden number problem instance where $m$ is the hidden number.

Naturally, we want to know how large $\ell$ needs to be for the HNP instance to be solvable. As in Section 3.3.1, the short vector we hope to recover by solving the HNP instance has length less than $\sqrt{\ell+2}B$. From Theorem 2.6, a shortest vector will have length approximately $\sqrt{\ell+2}(B^2 N^{\ell-1})^{1/(\ell+2)}$. Thus, we may expect LLL to successfully solve the HNP if $\sqrt{\ell+2}B$ is much less than this:

$$\begin{aligned} \sqrt{\ell+2}B &\ll \sqrt{\ell+2}(B^2 N^{\ell-1})^{1/(\ell+2)} \\ \implies B &\ll (B^2 2^{l(\ell-1)})^{1/(\ell+2)} \\ \implies 2^{l-16} &\ll (2^{2l-32} 2^{l(\ell-1)})^{1/(\ell+2)} \\ \implies l - 16 &\ll \frac{2l - 32 + l(\ell-1)}{\ell+2} \\ \implies l - 16 &\ll \frac{l + l\ell - 32}{\ell+2} \\ \implies (l-16)(\ell+2) &\ll l + l\ell - 32 \\ \implies l\ell - 16\ell + 2l - 32 &\ll l + l\ell - 32 \\ \implies l &\ll 16\ell \end{aligned}$$

For a $l = 512$ bit modulus, the lattice attack works when $\ell > 40$.

## References


[Adl83] Leonard M. Adleman. "On Breaking Generalized Knapsack Public Key Cryptosystems". In: *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*. STOC '83. New York, NY, USA: Association for Computing Machinery, 1983, pp. 402–412. ISBN: 0897910990. URL: https://doi.org/10.1145/800061.808771.

[Ajt98] Miklós Ajtai. "The shortest vector problem in $L_2$ is NP-hard for randomized reductions (extended abstract)". In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*. ACM Press, 1998. URL: https://doi.org/10.1145/276698.276705.

[Ara+20] Diego F. Aranha et al. "LadderLeak: Breaking ECDSA with Less than One Bit of Nonce Leakage". In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 225–242. ISBN: 9781450370899. URL: https://doi.org/10.1145/3372297.3417268.

[Bab86] L. Babai. "On Lovász' lattice reduction and the nearest lattice point problem". In: *Combinatorica* 6.1 (Mar. 1986), pp. 1–13. URL: https://doi.org/10.1007/bf02579403.

[BD99] Dan Boneh and Glenn Durfee. "Cryptanalysis of RSA with Private Key $d$ Less than $N^{0.292}$". In: *Advances in Cryptology — EUROCRYPT '99*. Ed. by Jacques Stern. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 1–11. ISBN: 978-3-540-48910-8.

[Ben+14] Naomi Benger et al. ""Ooh Aah... Just a Little Bit" : A Small Amount of Side Channel Can Go a Long Way". In: *Advanced Information Systems Engineering*. Springer Berlin Heidelberg, 2014, pp. 75–92. URL: https://doi.org/10.1007/978-3-662-44709-3_5.

[BH19] Joachim Breitner and Nadia Heninger. "Biased Nonce Sense: Lattice Attacks Against Weak ECDSA Signatures in Cryptocurrencies". In: *Financial Cryptography and Data Security*. Springer International Publishing, 2019, pp. 3–20. URL: https://doi.org/10.1007/978-3-030-32101-7_1.

[Ble98] Daniel Bleichenbacher. "Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1". In: *Advances in Cryptology — CRYPTO '98*. Springer Berlin Heidelberg, 1998, pp. 1–12. URL: https://doi.org/10.1007/bfb0055716.

[BM03] Johannes Blömer and Alexander May. "New Partial Key Exposure Attacks on RSA". In: *Advances in Cryptology - CRYPTO 2003*. Springer Berlin Heidelberg, 2003, pp. 27–43. ISBN: 978-3-540-45146-4.

[BV96] Dan Boneh and Ramarathnam Venkatesan. "Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes". In: *Advances in Cryptology — CRYPTO '96*. Springer Berlin Heidelberg, 1996, pp. 129–142. URL: https://doi.org/10.1007/3-540-68697-5_11.

[Cop96] Don Coppersmith. "Finding a Small Root of a Univariate Modular Equation". In: *Proceedings of the 15th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT'96. Springer-Verlag, 1996, pp. 155–165. ISBN: 354061186X.

[Cop97] Don Coppersmith. "Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities". In: *Journal of Cryptology* 10.4 (Sept. 1997), pp. 233–260. URL: https://doi.org/10.1007/s001459900030.

[Cor07] Jean-Sébastien Coron. "Finding Small Roots of Bivariate Integer Polynomial Equations: A Direct Approach". In: *Proceedings of the 27th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 379–394. ISBN: 3540741429.

[Cos+92] Matthijs J. Coster et al. "Improved low-density subset sum algorithms". In: *Computational Complexity* 2.2 (June 1992), pp. 111–128. URL: https://doi.org/10.1007/bf01201999.

[CR88] B. Chor and R.L. Rivest. "A knapsack-type public key cryptosystem based on arithmetic in finite fields". In: *IEEE Transactions on Information Theory* 34.5 (Sept. 1988), pp. 901–909. URL: https://doi.org/10.1109/18.21214.

[Ern+05] Matthias Ernst et al. "Partial Key Exposure Attacks on RSA up to Full Size Exponents". In: *Advances in Cryptology – EUROCRYPT 2005*. Springer Berlin Heidelberg, 2005, pp. 371–386. ISBN: 978-3-540-32055-5.

[Fri+88] Alan M. Frieze et al. "Reconstructing Truncated Integer Variables Satisfying Linear Congruences". In: *SIAM Journal on Computing* 17.2 (Apr. 1988), pp. 262–280. URL: https://doi.org/10.1137/0217016.

[Gal12] Steven D. Galbraith. *Mathematics of public key cryptography*. Cambridge University Press, 2012.

[HR07] Martin Hlaváč and Tomáš Rosa. "Extended Hidden Number Problem and Its Cryptanalytic Applications". In: *Selected Areas in Cryptography*. Springer Berlin Heidelberg, 2007, pp. 114–133. URL: https://doi.org/10.1007/978-3-540-74462-7_9.

[HS01] N. A. Howgrave-Graham and N. P. Smart. "Lattice Attacks on Digital Signature Schemes". In: *Designs, Codes and Cryptography* 23.3 (2001), pp. 283–290. URL: https://doi.org/10.1023/a:1011214926272.

[JM06] Ellen Jochemsz and Alexander May. "A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants". In: *Advances in Cryptology – ASIACRYPT 2006*. Springer Berlin Heidelberg, 2006, pp. 267–282. URL: https://doi.org/10.1007/11935230_18.

[Kal98] Burt Kaliski. *PKCS #1: RSA Encryption Version 1.5*. RFC 2313. Mar. 1998. URL: https://www.rfc-editor.org/info/rfc2313.

[Kan87] Ravi Kannan. "Minkowski's Convex Body Theorem and Integer Programming". In: *Mathematics of Operations Research* 12.3 (Aug. 1987), pp. 415–440. URL: https://doi.org/10.1287/moor.12.3.415.

[KJR16] B. Kaliski, J. Jonsson, and A. Rusch. *PKCS #1: RSA Cryptography Specifications Version 2.2*. Tech. rep. Nov. 2016. URL: https://doi.org/10.17487/rfc8017.

[KLL84] R. Kannan, A. K. Lenstra, and L. Lovász. "Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers". In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing - STOC '84*. ACM Press, 1984. URL: https://doi.org/10.1145/800057.808681.

[LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovász. "Factoring polynomials with rational coefficients". In: *Mathematische Annalen* 261.4 (Dec. 1982), pp. 515–534. URL: https://doi.org/10.1007/bf01457454.

[LO85] J. C. Lagarias and A. M. Odlyzko. "Solving low-density subset sum problems". In: *Journal of the ACM* 32.1 (Jan. 1985), pp. 229–246. URL: https://doi.org/10.1145/2455.2461.

[map22] maple3142. *TSJ CTF 2022 - Signature*. 2022. URL: https://github.com/maple3142/My-CTF-Challenges/tree/master/TSJ%20CTF%202022/Signature.

[May03] Alexander May. "New RSA vulnerabilities using lattice reduction methods." PhD thesis. Paderborn University, 2003.

[MH78] R. Merkle and M. Hellman. "Hiding information and signatures in trapdoor knapsacks". In: *IEEE Transactions on Information Theory* 24.5 (Sept. 1978), pp. 525–530. URL: https://doi.org/10.1109/tit.1978.1055927.

[Ngu09] P. Q. Nguyen. "Public-Key Cryptanalysis". In: *Recent Trends in Cryptography*. Ed. by I. Luengo. Vol. 477. Contemporary Mathematics. AMS–RSME, 2009.

[NS06] Phong Q. Nguyen and Damien Stehlé. "LLL on the Average". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006, pp. 238–256. URL: https://doi.org/10.1007/11792086_18.

[Odl91] A. M. Odlyzko. *The rise and fall of knapsack cryptosystems*. 1991. URL: https://doi.org/10.1090/psapm/042/1095552.

[PZ16] Yanbin Pan and Feng Zhang. "Solving low-density multiple subset sum problems with SVP oracle". In: *Journal of Systems Science and Complexity* 29.1 (Feb. 2016), pp. 228–242. URL: https://doi.org/10.1007/s11424-015-3324-9.

[Ron+19] Eyal Ronen et al. "The 9 Lives of Bleichenbacher's CAT: New Cache ATtacks on TLS Implementations". In: *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2019. URL: https://doi.org/10.1109/sp.2019.00062.

[SGM10] Santanu Sarkar, Sourav Sen Gupta, and Subhamoy Maitra. "Partial Key Exposure Attack on RSA – Improvements for Limited Lattice Dimensions". In: *Progress in Cryptology - INDOCRYPT 2010*. Springer Berlin Heidelberg, 2010, pp. 2–16. URL: https://doi.org/10.1007/978-3-642-17401-8_2.

[Sun+21] Chao Sun et al. “Guessing Bits: Improved Lattice Attacks on (EC)DSA with Nonce Leakage”. In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (Nov. 2021), pp. 391–413. URL: https://doi.org/10.46586/tches.v2022.i1.391-413.

[Wie90] Michael J. Wiener. “Cryptanalysis of short RSA secret exponents”. In: *IEEE Transactions on Information Theory* 36 (1990), pp. 553–558.

[Yas07] Takeshi NASAKO Yasuyuki MURAKAMI. *Knapsack Public-Key Cryptosystem Using Chinese Remainder Theorem*. Cryptology ePrint Archive, Report 2007/107. https://ia.cr/2007/107. 2007.