BRNO UNIVERSITY OF TECHNOLOGY

Faculty of Electrical Engineering
and Communication

BACHELOR'S THESIS

Brno, 2022

Martin Sečkár

BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

WEB APPLICATION DEMONSTRATING LATTICE-BASED CRYPTOGRAPHY

WEBOVÁ APLIKACE DEMONSTRUJÍCÍ KRYPTOGRAFII ZALOŽENOU NA MŘÍŽKÁCH

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR AUTOR PRÁCE	Martin Sečkár
SUPERVISOR VEDOUCÍ PRÁCE	M.Sc. Sara Ricci, Ph.D.

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Bachelor's Thesis

Bachelor's study program **Information Security**

Department of Telecommunications

Student:	Martin Sečkár	***ID:***	203640
Year of study:	3	***Academic year:***	2021/22

TITLE OF THESIS:

Web application demonstrating lattice-based cryptography

INSTRUCTION:

The assignment is focused on the development of a web application on lattice-based cryptography. At first, the student will study lattice-based cryptography (e.g. learning with error problem, shortest vector problem, Babai algorithm). A Part of the work will be the implemention of a web application that allows visualizing lattice, performing lattice computations, and basic lattice-based cryptographic protocols (e.g. Fiat-Shamir with abort signature). The writing of a manual is also required to the student.

RECOMMENDED LITERATURE:

[1] Bernstein, D.J., Buchmann, J., Dahmen, E.: Post-Quantum Cryptography. Springer (2008)

[2] Hoffstein, J., Pipher, J. C., Silverman, J. H., Silverman, J. H.: An introduction to mathematical cryptography. New York: springer (2008).

Date of project specification:	7.2.2022	***Deadline for submission:***	31.5.2022

Supervisor: M.Sc. Sara Ricci, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
Chair of study program board

ABSTRACT

The aim of this thesis is to develop and implement a web application demonstrating lattice-based cryptography. The application was developed using mainly the Python programming language and Docker container platform. More specifically, the modules utilize the Bokeh library and custom JavaScript functionality expanding the Bokeh library. The modules are hosted on a Flask server where the background calculations are being computed using numPy library. The application contains three modules describing the closest vector problem, learning with errors problem and the Boyen cryptographic protocol based on the latter problem. Users are able to visualize two dimensional lattices and perform selected computations. The codebase is easily expandable and can serve as a learning platform. The thesis also includes installation and user manual.

KEYWORDS

Babai's algorithm, Bokeh, data visualization, encryption, JavaScript, Lattice based cryptography, Learning with errors, protocol, Python

ABSTRAKT

Zámer tejto práce je vyvinúť a implementovať webovú aplikáciu demonštrujúcu kryptografiu založenú na mriežkach. Aplikácia bola vyvinutá použitím programovacieho jazyku Python a kontajnerizačnej platformy Docker. Špecifickejšie, implementované moduly používajú knižnicu Bokeh a vlastnú JavaScript funkcionalitu, ktorá rozširuje danú knižnicu Bokeh. Tieto moduly sú poskytované serverom Flask, kde taktiež prebiehajú všetky výpočty pomocou knižnice numPy. Aplikácia obsahuje tri moduly popisujúce problém najbližšieho vektora, problém učenia s chybami a Boyenov kryptografický protokol založený na predchádzajúcom probléme. Užívatelia majú možnosť vizualizovať dvojdimenzionálne mriežky a prevádzať vybrané výpočty. Zdrojový kód je jednoducho rozšíriteľný a môže slúžiť ako náučná platforma. Práca taktiež obsahuje inštalačný a používateľský manuál.

KĽÚČOVÉ SLOVÁ

Babaiov algoritmus, Bokeh, dátová vizualizácia, šifrovanie, JavaScript, kryptografia založená na mriežkach, učenie s chybami, protokol, Python

ROZŠÍRENÝ ABSTRAKT

Zadaním bakalárskej práce bolo vyvinúť webovú aplikáciu demonštrujúcu kryptografiu založenú na mriežkach. Na základe štúdia danej problematiky bola vyvinutá aplikácia v programovacom jazyku Python s použitím kontajnerovej platformy Docker. Táto aplikácia umožnuje užívateľom vizualizovať dvojdimenzionálne mriežky a prevádzať vizuálne operácie ako kalkuláciu Babai algoritmu. Ďalej aplikácia vizualizuje podobnosť medzi tažkými problémami na mriežkach, ako problémom najbližšieho vektora (CVP) a problémom učenia s chybami (LWE). Na koniec, aplikácia predstaví Boyenov kryptografický protokol založený na probléme učenia s chybami. Táto aplikácia slúži ako didaktický a vizualizačný nástroj použiteľný na vysvetlenie abstraktnej problematiky okolo kryptografie založenej na mriežkach. Jej vývoj a architektúra sú prispôsobené na efektívne rozšírenie o ďalšie moduly popisujúce ďalšie problematiky v post-kvantovej kryptografií. Architektúra aplikácie umožnuje rapídne spustenie buď lokálne na vlastnom zariadení napríklad pomocou aplikácie Docker Desktop, alebo na verejnom cloud hostingu.

Teoretická časť sa zaoberá vysvetlením implementovaných modulov. Predstavuje základné koncepty mriežkových štruktúr a ich báz. Je predstavený faktor ortogonality, tiež nazývaný Hadamardov pomer, ktorý vyjadruje ako su báze na seba kolmé. Ďalej sú predstavené výpočtovo ťažké problémy na mriežkach. Hlavný problém, ktorý je prezentovaný je problém najlbližšieho vektora. Na to je prezenotvaný Babaiov algoritmus, ktorý je schopný riešiť problém najbližšieho vektora ak sú báze mriežky dostatočne ortogonálne. Kapitola pokračuje základmi teórie pravdepodobnosti, ako definovaním náhodnej premennej a distribučnej funkcie, ktoré sú použité v nasledujúcej teórií. Distribučné funkcie su demonštrované na tisíc vzorkách výberu z danej distribúcie. Ďalej je zmienený problém učenia s chybami, ktorý je prirovnaný ku bežnému riešeniu sústav rovníc pomocou Gaussovej eliminačnej metódy. Na rozdiel od klasických sústav rovníc, učenie s chybami, ako názov napovedá, obsahuje chybový vektor, vďaka ktorému sa riešenie stáva výpočtovo tažké. Tento fakt je využiteľný ako kryptografická primitíva. Na dôkaz dostatočnej zložitosti je tento problém následne porovnaný s problémom najbližšieho vektora, kde je viditeľné, že je do tohto problému redukovateľný. Na koniec je predstavený Boyenov kryptografický protokol pre jeden bit, ktorý využíva problém učenia s chybami ako jeho kryptografickú primitívu. Je podrobne popísaný odporúčaný výber parametrov, šifrovanie a dešifrovanie.

Kapitola implementačné pozadie sa zaoberá okrem iného predstavením základov programovacieho jazyku Python. Ďalej prezentuje niekoľko dôležitých knižníc použitých pri vyvoji danej aplikácie, špecificky numPy, Bokeh a Flask. Je kladený dôraz na separáciu klientovej a serverovej strany programu, kde sa nachádzajú dva

servery obsiahnuté v kontajneri. Táto kapitola taktiež priblíži jazyk JavaScript, ktorý bol použitý na rozšírenie funkcionality knižnice Bokeh. Taktiež je predstavený formát JSON, ktorý je využitý ako výstupný dátový formát pre modul Boyenov kryptografický protokol. Na koniec je prezentovaná kontajnerizácia ako koncept pri vývoji cloud softvéru.

Praktická časť sa zaoberá implemetáciou aplikácie. Začína popisom zvolenej architektúry a technologického balíka. Potom popisuje vývojový proces cloudových aplikácií, ako sú CI/CD technológie a verzovací systém Git. Ďalej vysvetluje ako bola využitá knižnica numPy pre maticové výpočty. Následne je predstavená Bokeh knižnica a jej dátový model, view dekorátor a Jinja šablóny. Bokeh knižnica je primárne zameraná na vývoj interaktívnych grafov použitím serverovej technológie a callback funkcií, ktoré su ďalej demonštrované. Takisto sú predstavené JavaScript callback funkcie, ktoré slúžia ako rozšírenie základnej funkcionality knižnice Bokeh. Predstavené sú vykreslovacie schopnosti LaTeX objektov tejto knižnice a krátky návod ako rozšíriť aplikáciu o nové moduly. Ďalej je demonštrované vytvorenie Docker kontajneru pomocou Dockerfile súboru. Taktiež je spomenuté ako táto aplikácia využíva paralelné vlákna na spustenie Flask a Bokeh serveru. Predložená je aj jednoduchá bezpečnostná analýza, kde sú spomenuté potencionálne útoky na aplikáciu a ich možná mitigácia. Ďalej je samostatne popísané užívateľské rozhranie každého modulu spolu s užívateľským postupom. Na koniec je spomenutá inštalácia kontajnera a umiestnenie kontajnera na verejne prístupný cloud. Taktiež je vysvetlený postup možnej lokálnej inštalácie.

SEČKÁR, Martin. *Web application demonstrating lattice-based cryptography*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications, 2022, 70 p. Bachelor's Thesis. Advised by M.Sc. Sara Ricci, Ph.D.

Author's Declaration

Author:	Martin Sečkár
Author's ID:	203640
Paper type:	Bachelor's Thesis
Academic year:	2021/22
Topic:	Web application demonstrating lattice-based cryptography

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....

author's signature[*]

[*]The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank the advisor of my thesis, M.Sc. Sara Ricci, Ph.D. for her valuable comments, methodology advice, research insights and willingness to continue to work with me. I would also like to thank my family and friends for their perseverance through the easy and difficult times.

Contents

List of Figures

Listings

Introduction

Cryptography is an essential and almost invisible part of everyday life. It is one of those underlying technologies which general population presumes to be just working. With several historical breaches, whether by design or by continuous research, the notion of secure communication is slowly fading away. The former is regarding the BSAFE toolkit [1] proposed by RSA Security, a highly respected body in cybersecurity, which contained deliberately engineered backdoor in the pseudo-random number generator by National Security Agency (NSA) and the latter is regarding the finding of collisions in the MD5 algorithm [2]. These events can be seen as a motivation to develop mathematically safe cryptosystems.

The methods in cryptography came a long way from substitution ciphers used from around 500 BC to today's elliptic curve cryptography. This thesis is concerned with the future of cryptography methods, called post-quantum cryptography [3] (also quantum-resistant cryptography). Ever since the announcement of the Shor's algorithm, mathematicians predicted that some cryptographic primitives could be broken using quantum computers, hence the name post-quantum. Notably, cryptosystems based on integer factorization problem could be broken by calculating private keys using Shor's algorithm implemented on a large scale quantum computer. Although there are several questions whether such computer will be ever built, in 2017 the US National Institute of Standards and Technology (NIST) [4] has initiated a process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms. With recent announcements of IBM breaking the 100 qubit barrier [5], the threat of implementing sufficiently large scale Shor's algorithm is still low. Moreover, a significant number of experts and practitioners believe that a sufficiently powerful quantum computer can be built in the next decade. [6]

Lattice-based cryptosystems are being implemented as encryption schemes such as NTRUEncrypt [7], signatures (Dilithium [8]) or key exchange such as Saber [9]. Pradhan et al. [10] postulates that the Internet-of-Things (IOT) devices are a possible area of implementation since the need for storing large amount of keys is not satisfied by classic, e.g., not lightweight implementations of current cryptosystems. They further claim several other areas of interests, such as end-to-end encryption, electronic money and disk encryption. There have been several studies dealing with the use of lattice-based cryptography as a replacement for elliptic curves in blockchain technology [11], but the current status of these algorithms are still not able to outweigh the negatives in doing so. This ongoing research prompts us to create an accessible tool to spread the base knowledge of lattice-based algorithms to excite potential future research endeavors. We believe that if there is an easy and approachable way to learn this potentially obscure topic it could contribute to the greater academic interest.

The main goal of this thesis is to create a comprehensive learning platform where users can easily and visually learn about the underlying mathematics of lattices and problems on the lattices with emphasis on user-friendly interface and easy deployment. For these reasons we believe the best platform to be a web application. We will also show the reasoning behind selecting Python programming language along its extensive library collection as our main platform.

We describe selected lattice notions necessary for the reader to comprehend the basics of the hard problems associated with lattices. We show how bases generates lattice, the need for bases to be linearly independent, how lattices can be generated from many bases and how to distinguish between them. We have decided to show the Closest Vector Problem (CVP), since the Learning with errors problem is reducible to CVP. Furthermore we show the Babai's Closest Vertex algorithm to describe how can CVP be solved when using insecure bases.

We propose a solution based on chosen architecture which encompasses all the necessary prerequisites and show basic security hardening of the application. Furthermore we describe how we have deployed the application on a public cloud using Docker. Moreover, we show the local deployment options with Docker. At the end, we describe the user interface and the workflow the user will apply to do basic visual calculations.

1 Background

In this chapter, we lay out the need for post-quantum cryptography, the theory of lattices as mathematical structures. Some linear algebra knowledge is presumed to fully comprehend the mathematical background, such as vector and matrix notation and multiplication. We continue with describing elementary knowledge of probability theory, the notation and distribution functions. Moreover we describe the basics of learning with errors problem, the connection to lattice problems and a show a protocol using the aforementioned problem as a cryptographic primitive.

1.1 Post-Quantum Cryptography

With the emergent research of quantum computing, the risk of breaking cryptographic primitives based on discrete logarithm and/or integer factorization problem are imminent [12]. For this reason, mathematicians are trying to find other hard problems, i.e., Nondeterministic Polynomial time (NP)-hard problems that can be used as a basis for new post-quantum era cryptosystems. One of the biggest and well-supported approaches is the use of hard problems on lattices.

1.2 Linear Independence

To understand the definition of a lattice, we need to define a vector set property called linear independence. The vectors are linearly independent if they are not a linear combination of each other i.e. if there is only one solution to the equation

$$a_1\mathbf{b}_1 + ... + a_n\mathbf{b}_n = 0 \tag{1.1}$$

and that is $a_j = 0$, where $j \in \mathbb{Z}$.

1.3 Basis

In general, a vector set $\mathbf{B}$ in a vector space $\mathbf{V}$ is a basis, if $\mathbf{B}$ is linearly independent and span $\mathbf{V}$, which means all elements in $\mathbf{V}$ can be represented as a linear combination of $\mathbf{B}$.

1.4 Lattices

Lattice [3] is a closed set of all integer multiples of basis vectors. Generally it is defined as

$$\mathcal{L}\{\mathbf{b}_1, ..., \mathbf{b}_n\} = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i : x \in \mathbb{Z} \right\}, \tag{1.2}$$

where $\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{R}^n$ are linearly independent vectors called the basis and n is the number of dimensions.
For the purpose of this thesis, we will use the matrix form of the basis, which is constructed as

$$\mathbf{B} = \begin{pmatrix} | & ... & | \\ \mathbf{b}_1 & ... & \mathbf{b}_n \\ | & ... & | \end{pmatrix}. \tag{1.3}$$

Matrix $\mathbf{B}$ is in some literature called the generator of the lattice $\mathcal{L}$. Using the matrix form, Equation 1.2 becomes

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^n\}. \tag{1.4}$$

Note that the same lattice has infinitely many generators, which is a useful property we will show later.

1.4.1 Example

Let lattice $\mathcal{L}$ be defined as linear combination of generator matrix

$$\mathbf{B} = \begin{pmatrix} 2 & 1 \\ 0 & 3 \end{pmatrix}. \tag{1.5}$$

Based on Equation 1.4, lattice $\mathcal{L}$ can be written as an infinite set of integer points, such as

$$\mathcal{L} = \{(1, 28), (2, 31), (3, 34), (4, 37), (1, 23), (2, 26), (3, 29), (4, 32), (5, 35), ...\} \tag{1.6}$$

The lattice $\mathcal{L}$ can be partially seen in Figure 1.1 with red arrows showing the basis vectors.

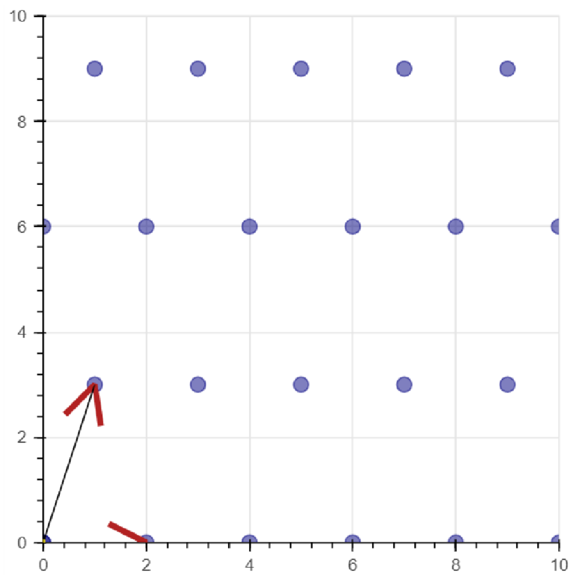


Fig. 1.1: Example of lattice with highlighted basis.

1.5 Hadamard Ratio

To distinguish between these bases (generators), we use the Hadamard ratio [13], which can be seen as the orthogonality factor. It is defined as

$$\mathcal{H}(\mathcal{L}) = \left(\frac{|\det \mathcal{L}|}{\|\mathbf{b}_1\|_2 \cdot \ldots \cdot \|\mathbf{b}_n\|_2} \right)^{1/n}. \tag{1.7}$$

This coefficient determines if the bases are "good" or "bad", depending on the value. We consider a "good" basis to have $\mathcal{H}(\mathcal{L}) > 0.75$ and a "bad" basis as $\mathcal{H}(\mathcal{L}) < 0.25$.

1.5.1 Example

Using the lattice $\mathcal{L}$ from previous example, we get

$$\mathcal{H}(\mathcal{L}) = \left(\frac{6}{2 \cdot \sqrt{10}} \right)^{1/2} \simeq 0.974. \tag{1.8}$$

We will consider this as a "good" basis.

1.6 Unimodular Martix

Unimodular matrix $\mathbf{A}$ is a matrix which satisfies the equation

$$\det \mathbf{A} = \pm 1, \tag{1.9}$$

The inverse of matrix $\mathbf{A}$ is another unimodular matrix. We will use a random unimodular matrix to create a "bad" basis out of a "good" one i.e. a basis with lower Hadamard ratio. Note that based on the properties of unimodular matrices, the basis created by applying unimodular matrix will be generating the same lattice as the original one.

1.6.1 Generating unimodular matrix

The generation of a random unimodular matrix is conditioned by Equation 1.9. This is achieved by generating two matrices with ± 1 on the diagonals and a random triangular matrix under or over the diagonal. The result of multiplying these matrices is always a matrix with det $= \pm 1$. Mathematically, this calculation can be shown on example as

$$\det \mathbf{U} = \det \left[\begin{pmatrix} 1 & -3 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 2 & -1 \end{pmatrix} \right] = \det \begin{pmatrix} -7 & 3 \\ 2 & -1 \end{pmatrix} = 1. \qquad (1.10)$$

1.7 q-ary Lattice

In cryptography, q-ary lattices are the most interesting, as they are in one-to-one correspondence with linear codes $\mathbb{Z}_q^n$. Lattice $\mathcal{L}$ embedded in $\mathbb{Z}^n$ is a q-ary lattice for an integer q, if $q\mathbb{Z} \subseteq \mathcal{L}$. That means the vector $\mathbf{b}$ is a member of $\mathcal{L}$ if and only if $\mathbf{b} \mod q$ is also a member of $\mathcal{L}$. With this knowledge, we can rewrite Equation 1.4 as

$$\mathcal{L}(\mathbf{B}) = \{\mathbf{Bx} \mod q : \mathbf{x} \in \mathbb{Z}^n\}. \qquad (1.11)$$

1.8 Closest Vector Problem

There are several NP-hard problems defined on lattice $\mathcal{L}$, namely Shortest Vector problem (SVP), Shortest Independent Vectors Problem (SIVP) and Closest Vector problem (CVP). There are polynomial time reduction algorithms which allow transform of one problem into another. Moreover, CVP was shown by van Emde Boas to be NP-hard [14]. Since the SVP can be generalized into CVP, we will focus on the latter. Closes vector problem (CVP) states that given a lattice generating matrix $\mathbf{B}$ and a target vector $\mathbf{t}$, it is hard to find the closest vector $\mathbf{v} \in \mathcal{L}$. Formally

$$dist(\mathcal{L}, \mathbf{t}) = \min_{\mathbf{x} \in \mathcal{L}} \|\mathbf{x} - \mathbf{v}\|, \qquad (1.12)$$

where *dist* is a distance function.

1.9 Babai's Closest Vertex Algorithm

Let $\mathcal{L} \in \mathbb{R}^n$ be a lattice with basis $\mathbf{B}$ and let $\mathbf{t} \in \mathbb{R}^n$ be an arbitrary vector. If the vectors in the basis are sufficiently orthogonal to one another, then the following algorithm solves CVP.

Algorithm 1 Babai's Closest Vertex Algorithm

```
$\mathbf{t} \leftarrow t_1 b_1 + ... + t_n b_n$, where $t_1, ..., t_n \in \mathbb{R}$
$\mathbf{a}_i \leftarrow \lceil \mathbf{t}_i \rfloor$, for $i = 1, 2, ..., n$
    return $\mathbf{v} = a_1 b_1 + a_2 b_2 + ... + a_n b_n$
```

1.9.1 Example

Using lattice defined in previous examples, let's say we are looking for the closest member vector to vector $\mathbf{t} = (2.5, -3.2)$. Using Algorithm 1, we start by defining a vector $\mathbf{t}$ as

$$\mathbf{t} = \begin{pmatrix} 2.5 \\ -3.2 \end{pmatrix} = t_1 \begin{pmatrix} 2 \\ 0 \end{pmatrix} + t_2 \begin{pmatrix} 1 \\ 3 \end{pmatrix}. \tag{1.13}$$

Now we need to solve for the unknown variables t_n

$$t_2 = -\frac{3.2}{3} = -1.067; t_1 = \frac{2.5 - \mathbf{t}_2}{2} = 1.7835, \tag{1.14}$$

and round the results to the nearest integer

$$a_2 = -1; a_1 = 2. \tag{1.15}$$

This gives us the closest vector

$$\mathbf{v} = 2\mathbf{b}_1 - \mathbf{b}_2 = \begin{pmatrix} 3 \\ -3 \end{pmatrix}. \tag{1.16}$$

Visually, this calculation is shown in Figure 1.2, where the yellow arrow represents vector $\mathbf{t}$ and orange arrow represents the closest vector $\mathbf{v}$ in lattice $\mathcal{L}$.

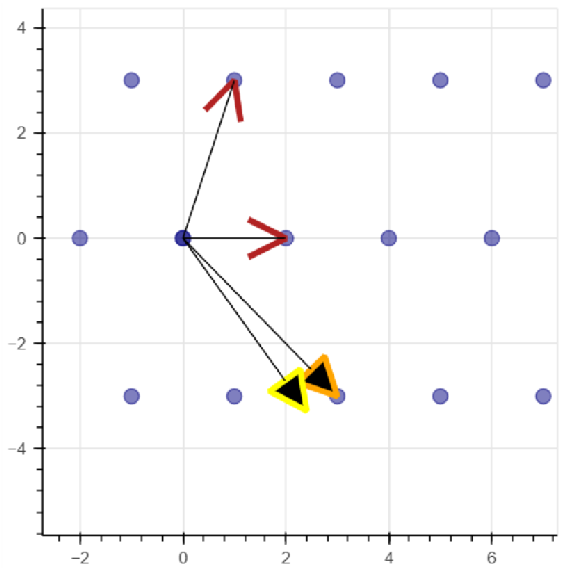


Fig. 1.2: Example of Babai's closest vertex algorithm.

1.10 Random variable

For the purposes of describing the probability distribution, one has to define the fundamental measure of probability. Probability [15] is measured using a random variable, which is a function in the form of

$$X : \Omega \to E, \tag{1.17}$$

where X is a random variable, Ω is a sample space and E is a measurable space.

A distribution of random variable is defined as

$$F_X(x) = P(X \leq x) = \sum_{x_i < x} P(X = x_i). \tag{1.18}$$

1.10.1 Example

In a fair game of coin toss, the coin has only two states: heads or tails. This can be represented as

$$X : \{h, t\} \to \{0, 1\}. \tag{1.19}$$

In the case of two rounds, the possible states increase to

$$X : \{hh, ht, th, tt\} \to \{00, 01, 10, 11\}. \tag{1.20}$$

Then the probabilities of tossing each pair are equal

$$P(X = hh) = P(X = ht) = P(X = th) = P(X = tt) = \frac{1}{4}. \tag{1.21}$$

1.11 Probability distribution

For the purposes of this thesis, we are concerned with two types of probability distribution, namely uniform and normal distribution. The former distribution has a form of a function

$$f(x) = \begin{cases} \frac{1}{b-a}, & \text{for } x \in [a, b] \\ 0 & \text{otherwise.} \end{cases} \tag{1.22}$$

The latter has a form of a function $\mathcal{N}(\mu, \, \sigma^2)$ with distribution

$$d_n(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{2\sigma^2}}, \tag{1.23}$$

where μ is the mean and σ is standard deviation.

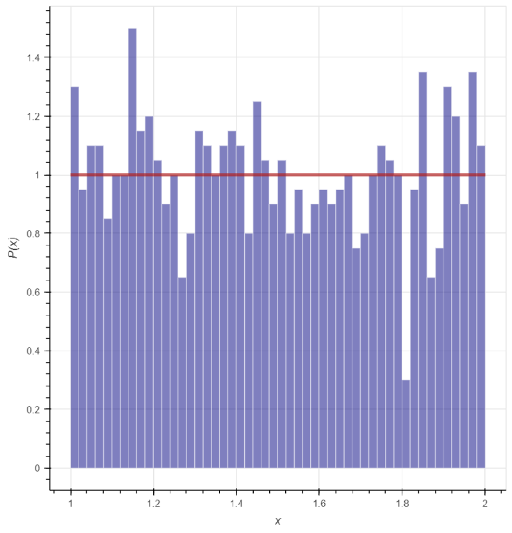


Fig. 1.3: Uniform distribution.

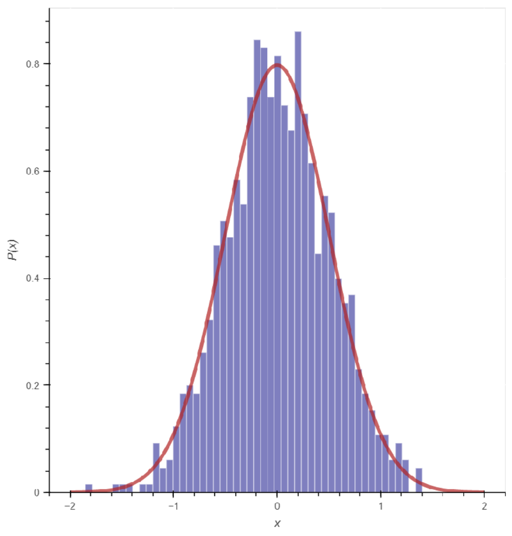


Fig. 1.4: Normal distribution.

Figure 1.3 shows the uniform distribution sampled 1000 times with parameters set to $a = 1, b = 2$. Figure 1.4 shows the normal distribution sampled 1000 times with parameters set to $\mu = 0, \sigma = 0.5$.

1.11.1 Example

Continuing the example shown in Equation 1.20, there are three possible outcomes of the two rounds of coin toss: either both coins are heads, both coins are tails or one is heads and one tails. This scenario is depicted in the measurable space as $E = \{0, 1, 2\}$, and the distribution function has values

$$\begin{aligned} F_X(0) &= P(X = hh) = \frac{1}{4}, \\ F_X(1) &= F_X(0) + P(X = ht) + P(X = th) = 3\frac{1}{4} = \frac{3}{4}, \\ F_X(2) &= F_X(1) + P(X = tt) = 4\frac{1}{4} = 1. \end{aligned} \tag{1.24}$$

Figure 1.5 depicts the distribution function of the game described above.

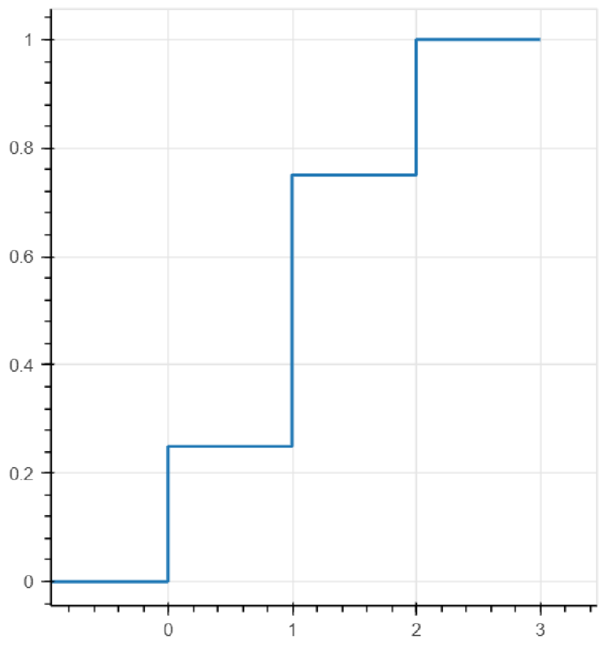


Fig. 1.5: Discrete distribution function of a two round coin toss.

1.12 Solving linear equations

Considering a set of equations in form of

$$\begin{aligned} a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 &= b_1 \mod p, \\ a_5x_1 + a_6x_2 + a_7x_3 + a_8x_4 &= b_2 \mod p, \\ a_9x_1 + a_{10}x_2 + a_{11}x_3 + a_{12}x_4 &= b_3 \mod p, \\ a_{13}x_1 + a_{14}x_2 + a_{15}x_3 + a_{16}x_4 &= b_4 \mod p, \end{aligned} \tag{1.25}$$

where we try to find $(x_1 ... x_n) \in \mathbb{Z}_p^n$. Using a matrix form, we obtain

$$\mathbf{A}x = \mathbf{B}(\mod p), \tag{1.26}$$

where $\mathbf{A}$ and $\mathbf{B}$ are known matrices and p is a known prime. Retrieval of the vector s is trivial using Gaussian elimination method in polynomial time.

1.12.1 Example

We take random samples from a prime group of order $p = 5$ and populate the matrices

$$\mathbf{A} = \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix}, \text{ and } \mathbf{B} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}. \tag{1.27}$$

Solving for x, using Gaussian elimination, we obtain

$$x = \begin{pmatrix} -7 \\ 5 \end{pmatrix} \mod 5 = \begin{pmatrix} 3 \\ 0 \end{pmatrix}. \tag{1.28}$$

1.13 Learning with errors

By slightly changing the Equation 1.25, we obtain

$$\begin{aligned} a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 &\sim b_1 \mod p, \\ a_5x_1 + a_6x_2 + a_7x_3 + a_8x_4 &\sim b_2 \mod p, \\ a_9x_1 + a_{10}x_2 + a_{11}x_3 + a_{12}x_4 &\sim b_3 \mod p, \\ a_{13}x_1 + a_{14}x_2 + a_{15}x_3 + a_{16}x_4 &\sim b_4 \mod p, \end{aligned} \tag{1.29}$$

where we try to find $(x_1 ... x_n) \in \mathbb{Z}_p^n$ and the equations are true up to a certain small error. Now the Gaussian elimination does not work anymore and the problem is considered to be hard to solve.

Again, using the matrix notation, an error vector e with given distribution is added. By adding an error vector, the equation is in the form

$$\mathbf{A}x + e = \mathbf{B}(\mod p), \tag{1.30}$$

Solving this equation without the knowledge of the error vector is provably NP-hard. This problem was formulated by Regev [16] and is known as the Learning With Errors (LWE) problem.

Definition 1 (LWE Problem) *Given $n.m, q \in \mathbb{Z}$ and $\mathcal{X}$, a distribution in $\mathbb{Z}_q$, with arbitrary many sample pairs $(A, As + e)$, where $A \in_{UR} \mathbb{Z}_q^{m\times n}$ and $e \in_{\mathcal{X}^m} \mathbb{Z}_q^n$, compute a vector $s \in_{UR} \mathbb{Z}_q^n$.*

Notation $\in_{UR}$ describes a uniform sampling from a given group and $\in_{\mathcal{X}^m}$ describes a sampling using the chosen distribution $\mathcal{X}$.

1.13.1 Example

Using parameters from Equation 1.27, and adding a random sampled error vector

$$e = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \tag{1.31}$$

we obtain equation

$$\begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} + \begin{pmatrix} -1 \\ 1 \end{pmatrix} \mod 5 \sim \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \tag{1.32}$$

which would be hard to solve without the knowledge of vector e.

1.13.2 Connection to lattice problems

Equation 1.30 can be thought of as a lattice based operation on a basis B and having the secret s as the closest vector shown in Figure 1.2. Moreover, what has been shown as a selected input vector to the Babai algorithm can be seen as the result $\mathbf{B}$ of the Equation 1.30. By transforming the LWE problem to the lattice, the hardness of LWE is similar to solving CVP. Figure 1.6 depicts the LWE solution and the public key on the lattice with highlighted basis. We can see the similarities to Figure 1.2.

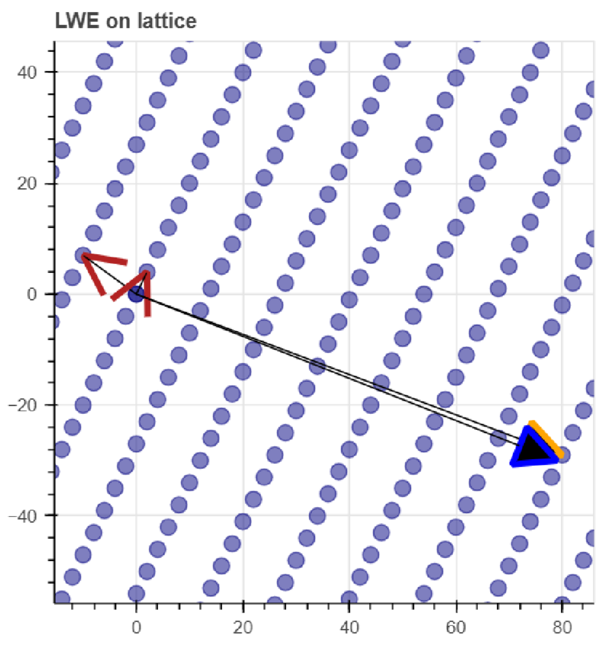


Fig. 1.6: Plotted LWE parameters seen as equivalent problem to CVP.

1.13.3 Applications

Current state of the art applications are utilizing the ring learning with errors variant, which proves to be less computationally demanding. The main difference is that the matrices are not populated by random group elements but with random polynomials on a defined ring. The main contenders for the post-quantum cryptography standards include NTRUSign [7], RLWE-SIG and Dilithium [8] for signature schemes and NTRUEncrypt and Saber [9] for encryption schemes, where Saber uses the Learning with rounding variant.

1.14 LWE encryption protocol

The protocol described by Boyen [17] is used to encrypt a single bit of information and serves as an example for a full encryption system based on LWE.

1.14.1 Chosing the parameters

To generate the keys, Alice selects a prime modulus q, a random matrix $\mathbf{A} \in_{UR} \mathbb{Z}_q^{n\times m}$ as shown in Equation, as well as a random vector $s \in_{UR} \mathbb{Z}_q^m$ and a random error sample vector $e \in_{\psi_\alpha} \mathbb{Z}_q^m$, where ψ_α is a chosen probability distribution function. Note that Chen et al. [18] recommend the dimension m to be at least $2n\log(q)$ and the prime q to be in the range $n^2 < q < 2n^2$. Furthermore the error distribution parameter α is set to be $\alpha = \frac{1}{\sqrt{n}\log^2(n)}$, which we have rounded to 1. That gives us a set of integer errors $\{-1, 0, 1\}$.

Using these parameters, the LWE problem is constructed as $p = \mathbf{A}s + e \mod q$. The matrices $\mathbf{A}$ and p are then used as a public key parameters that are sent to Bob and the error vector e is considered a private key.

1.14.2 Encryption

Bob selects a message $b \in \{0, 1\}$ and computes the Equations

$$\begin{aligned} Enc_{A,p}(b) = (a', p') = (&\sum_I a_i, \\ &\sum_I p_i + b\lfloor\frac{q}{2}\rfloor), \end{aligned} \tag{1.33}$$

where the first parameter is a row sum of the matrix $\mathbf{A}$ and the second parameter is a row sum of the vector p with the added bit message b multiplied by a floor function of the half of the prime modulus q. These vectors are then sent to Alice for decryption.

1.14.3 Decryption

Alice calculates Equation

$$e' = p' - a' \times s^T \mod q, \tag{1.34}$$

and decrypts the message according to

$$Dec_s(a'.p') = \begin{cases} 0, & \text{if } e' \sim 0 \\ 1, & \text{if } e' \sim \lfloor\frac{q}{2}\rfloor. \end{cases} \tag{1.35}$$

The decryption can be shown on a unit circle, where the modular group can be thought of as a clock. The uppermost point on the circle is equivalent to the "0" case in the Equation 1.35, and the lowermost point is equivalent to the "1" case, respectively. Then the decryption is visualized as a point in the neighborhood of these points.

1.14.4 Example

Suppose we have selected the parameters such that we have obtained the LWE problem as

$$\begin{pmatrix} -13 & 9 & 1 & 1 \\ 12 & 3 & -8 & 9 \\ -12 & -3 & 0 & 8 \end{pmatrix} \begin{pmatrix} 16 \\ -16 \\ 1 \\ -8 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} = \begin{pmatrix} 15 \\ 12 \\ 13 \end{pmatrix} \mod 17, \tag{1.36}$$

where the public parameters are shown in blue and private parameters are shown in red. Then the encryption of a bit $b = 1$ will give us

$$\begin{aligned} Enc_{A,p}(1) = (a', p') = (\sum_I \begin{pmatrix} -13 & 9 & 1 & 1 \\ 12 & 3 & -8 & 9 \\ -12 & -3 & 0 & 8 \end{pmatrix} = \begin{pmatrix} 4 \\ 9 \\ 10 \\ 1 \end{pmatrix}, \\ \sum_I \begin{pmatrix} 15 \\ 12 \\ 13 \end{pmatrix} + 1\lfloor \frac{17}{2} \rfloor = 14). \end{aligned} \tag{1.37}$$

Following the scheme, decryption starts with calculating the parameter

$$e' = 14 - \begin{pmatrix} 4 \\ 9 \\ 10 \\ 1 \end{pmatrix} \times \begin{pmatrix} 16 & -16 & 1 & -8 \end{pmatrix} \mod 17 = 7, \tag{1.38}$$

and finding the nearest point

$$Dec_s(a'.p') = \begin{cases} 0, & \text{if } 7 \sim 0 \\ 1, & \text{if } 7 \sim 8 \end{cases} = 1. \tag{1.39}$$

Figure 1.35 depicts the calculated value interpolated to a point on a unit circle and shows that the value is close to case "1".

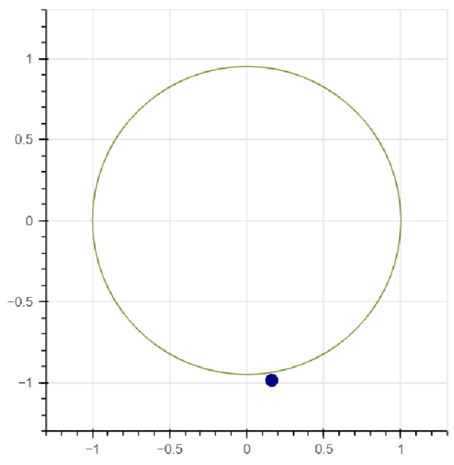


Fig. 1.7: Decryption of the equation e'.

Note that we have omitted the recommendation set by Chen et al. in order to save space, as the matrix with parameter $n = 4$ would have at least 10 dimensions.

2 Implementation Background

In this chapter, we introduce the Python programming language as a tool for data visualization. We take a look at some of the Python libraries used that allow easy linear algebra opertions on matrices, such as numPy, and we show how Flask routing functions work. Furthermore we describe the elementary knowledge of JavaScript and how it was utilized in this work. We will conclude with the high-level overview of the containerization as our deployment platform.

2.1 Python

Python [19] is an interpreted, multi-paradigm programming language which is highly popular amongst data science community. In this thesis, we will concern ourselves with Python version 3.9. We will look closer at numPy, Flask and Bokeh.

2.2 numPy

NumPy is the primary array programming library for the Python language. It has an essential role in research analysis pipelines in fields as diverse as physics, chemistry, astronomy, geoscience, biology, psychology, materials science, engineering, finance and economics [20]. At it's core lays the `array()` function, which creates the matrix model using nested arrays. For example, vector **B** in Equation 1.5 can be described by `numpy.array([[2,1],[0,3]])`.

2.3 Flask

Flask is a web microframework. Based on the Foreword [21], the "micro" in microframework means the core of the framework is simple but extensible. That means developers have the freedom to choose their own technology stack. It also supports Jinja template engine.

The most important functionality in Flask for us is the route decorator. This decorator put in front of a view function will execute the function if there will be a request made on the endpoint of the decorator.

2.3.1 Basic Flask Application

```
from flask import Flask

```

```python
app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello,␣World!</p>"
```

Listing 2.1: Hello world implemented in Flask

Listing 2.1 shows the use of the route decorator and view function returning a simple HTML page.

2.4 Bokeh

Bokeh is a Python library for creating interactive visualizations. [22] The introduction claims the developer is able to create JavaScript-powered visualizations with no need to write JavaScript, which is true for some cases such as this, but further modifications ie. more customized tools would require JavaScript code. The main reason this library was chosen is the ability to modify data sets and update the plots in real time. We believe the real time aspect in a learning tool is crucial for maximizing success of understanding the underlying mathematical concepts. We will focus on the server-side modules, namely bokeh.models and bokeh.plotting.

2.4.1 bokeh.models

Everything that comprises a Bokeh plot or application tools, controls, glyphs, data sources is a Bokeh Model. Bokeh models are configured by setting values their various properties [22]. These models are encompassed into a Document which can be serialized and displayed in browser using BokehJS library.

2.4.2 bokeh.plotting

The bokeh.plotting API is the primary interface and is centered around the figure() command and the associated glyph functions. [22]

2.4.3 Bokeh Server

The primary purpose of the Bokeh server is to synchronize data between the underlying Python environment and the BokehJS library running in the browser. Bokeh server is used to stream large data sets, or to enable complex user interactions based on widgets and selections. [22]

2.4.4 Basic Bokeh Application

Listing 2.2 shows the basic example of a static plot saved into a HTML file.

```
from bokeh.plotting import figure, output_file, show

# output to static HTML file
output_file("line.html")

p = figure(width=400, height=400)

# add a circle renderer with a size, color, and alpha
p.circle([1, 2, 3, 4, 5], [6, 7, 2, 4, 5], size=20, color
    ="navy", alpha=0.5)

# show the results
show(p)
```

Listing 2.2: Simple plot implemented in Bokeh

2.5 JavaScript

JavaScript is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. In this thesis we will concern ourselves minimally with JavaScript, as most of the client-side computations are conducted through Python API that already serializes objects into JavaScript. Our only requirement for this language is to write custom modules that are not at this time available in the Bokeh library.

2.5.1 JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. Bokeh library natively uses JSON to send and update data from server to client and vice versa. We will use this format as output from our custom download widget. The notation support nested parameters. Listing 2.3 depicts an example of JSON format.

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

Listing 2.3: Example of JSON format

2.6 Containerization

This technology allows developers to package software with minimized environment it was created for. This approach proves to be very helpful as it mitigates most of the issues associated with deployment. It can be seen as a lightweight alternative to virtualization. Containerization is the packaging of software code with just the Operating System (OS) libraries and dependencies required to run the code to create a single lightweight executable—called a container—that runs consistently on any infrastructure [23]. Docker is the most popular containerization platform today.

3 Implementation

In this section, we introduce the choices made that lead to finishing the application. The main goal of the application is to build a teaching tool explaining the mathematical concepts behind state of the art lattice-based cryptosystems. This was achieved by creating a vector set visualization using Bokeh and selected algorithms using numPy. We show what technologies were used to create the application, the reasoning behind selecting the technological stack, the overall architecture of the appliction and describe the utilization of numPy library for vector algebra operations and utilization of Bokeh Server as the real-time plotting tool. We also describe the process of developing custom JavaScript modules. Furthermore, we will lay out the interface of the application and the overall user interface. As one of the requirements was for the application to be deployable either locally or on a server, containerization with Docker was chosen as the best candidate.

3.1 Selection of technological stack

From the start, Python was the first the first choice, as it is the most popular programming language in the field of data science, mainly due to it's multi-paradigm philosophy and the vast variety of libraries, which are contributing to the ease of use. For these reasons, we have chosen Flask, Bokeh and numPy as our main platform for the service hosting, visualizations and calculations respectively. We have also chose Docker as the main distribution platform for its ease of use and out-of-the-box deployability. In the process of development, Continuous Integration/Continuous Development (CI/CD) tools, such as GitHub were utilized to ease the deployment and verification process.

3.2 Preparation of the development environment

When developing an application in Python, it is generally a good practice to create and work in a virtual environment. This allows the developer a closer control of the package dependencies. Listing 3.1 shows the basic steps to install and create virtual environment presuming the developer has pip already installed and is using a Unix shell.

```
pip install virtualenv #Installation of the virtualenv
   package
```

```
virtualenv venv #Creation of virtual environment named
    venv
source venv/bin/activate #This script will activate the
    environment
```

Listing 3.1: creation of virtual environmnet.

One of the use cases of using virtual environments is the ease to recreate the environment on any machine. This can be done by simply exporting the already existing dependencies into a text file, as shown in Listing 3.2.

```
pip freeze > requirements.txt
```

Listing 3.2: Exporting dependencies.

3.2.1 Git

As mentioned before, the project was developed with CI/CD techniques in mind, such as proper versioning and instant deployment. For this reason, GitHub platform was utilized, as it provides developers with the best solutions out of the box. Since the project was developed inside the virtual environment, it was very easy to upload the whole environment into the Git versioning system and continue the development from any device.

3.3 Architecture

At the core of the application, Flask is hosting the Web application on port 80 and Bokeh server has its endpoints on port 50007, where the webhooks are hosted. Flask is providing browsers with rendered Jinja template which contains the reference to Bokeh document. Bokeh server is on change updating the documents created in each session using the created webhooks by updating the serialized YAML data. Figure 3.1 depicts this use of the ports, and the creation of several user sessions.

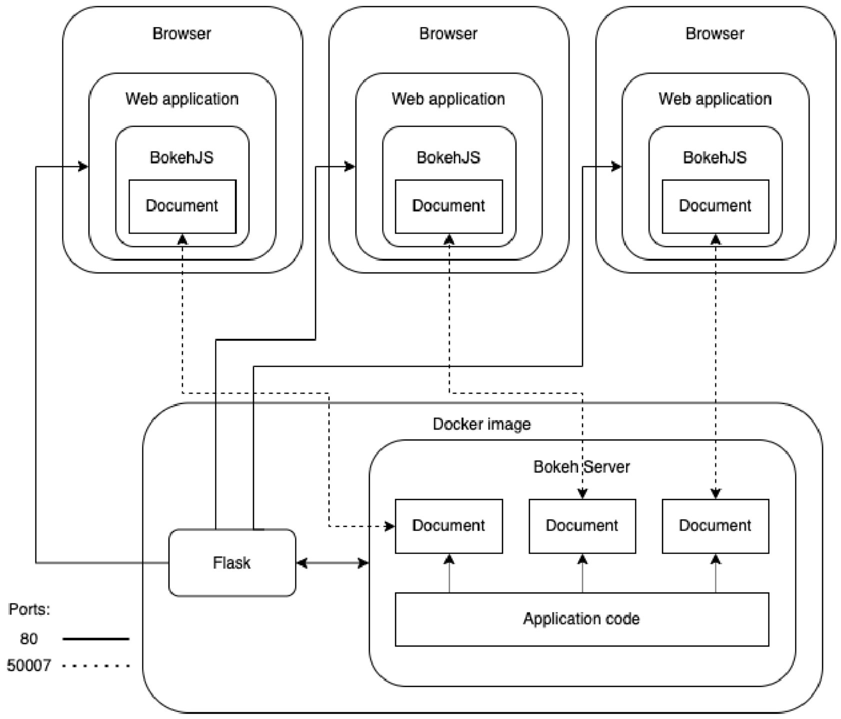


Fig. 3.1: Architecture of the application.

3.3.1 File Structure

```
crypto_lattice ...................................... Root folder
 └─ app ............................................. Application package
     ├─ __init__.py ............ Definition of the application package
     ├─ babai.py .......................... Custom Bokeh document
     ├─ protocol.py ....................... Custom Bokeh document
     ├─ lwe_basis.py ...................... Custom Bokeh document
     ├─ views.py .......................... Flask's view functions
     ├─ download.js ....................... JavaScript module
     ├─ templates ......................... Jinja templates
     │   ├─ index.html
     │   ├─ lwe.html
     │   └─ alg.html
     └─ static
         └─ style.css
 ├─ run.py .......................................... Starting script
 ├─ requirements.txt ................... Environment dependencies
 └─ Dockerfile ......................... Docker image template
```

Fig. 3.2: Directory tree of crypto_lattice repository.

Figure 3.2 shows the structure of the project. The file structure was chosen on purpose, as it provides clear space for future expansion. We will describe the functionality of each files in further sections.

3.3.2 Views

The application has several view functions, one for each module. These views route the Jinja template combined with the Bokeh server document onto one endpoint. The view depicted in Listing 3.3 is listening on root directory / where it is serving the index.html template with injected script referring to the `:50007/babai` endpoint using Bokeh's `server_document` function.

```python
@app.route('/', methods=['GET'])
def bkapp_page():
    script = server_document('http://localhost:50007/
       babai')
    return render_template("index.html", script=script,
       template="Flask")
```

Listing 3.3: Flask view function.

3.3.3 Jinja and rendering templates

Jinja is a web template engine for the Python programming language [25]. It utilizes special tags to render inputted objects. In our case we use the templates to add HTML objects from the Bootstrap framework and render the associated modules.

3.4 Vector algebra using numPy

One of the main reasons for choosing numPy as the computational library is the existence of `numpy.linalg` functions, which enables developers to easily and efficiently do calculations on matrices. For the purpose of this application, we will be referring to the generating basis in the code as a matrix in form of Equation 1.3.

3.4.1 Generating lattices

The function depicted in Listing 3.4 is a direct implementation of Equation 1.2, although not infinite for obvious computational limitations. We have omitted the use of modulus on the lattice, since the function does not generate sufficient number of points to affect the computations using an arbitrary prime modulus. This function was partially inspired by the script found at `asecuritysite.com`[1] and generally repurposed for the use with numPy.

```python
def generate_lattice(basis):
    ... # initialization of variables omitted
    for a in range(-50, 50):
        for b in range(-50, 50):
                xnew = a * basis[0][0] + b * basis[0][1]
                xval.append(xnew)
                ynew = a * basis[1][0] + b * basis[1][1]
                yval.append(ynew)
```

Listing 3.4: Lattice generator function.

3.4.2 Generating unimodular matrix

Listing 3.5 shows the calculation performed by example Equation 1.10.

```python
def rand_unimod(n):
    upperTri = np.triu([[np.random.randint(-3,3) for _ in
        range(n) ] for _ in range(n)],1)
    lowerTri = np.tril([[np.random.randint(-3,3) for _ in
        range(n) ] for _ in range(n)],-1)

    for r in range(len(upperTri)):
        for c in range(len(upperTri)):
            if(r==c): #Put either 1 or -1 on diagonal
                if bool(random.getrandbits(1)):
                    upperTri[r][c] = lowerTri[r][c] = 1
                else:
                    upperTri[r][c] = lowerTri[r][c] = -1

    return np.matmul(upperTri,lowerTri)
```

Listing 3.5: Random unimodular matrix function.

3.4.3 Calculating Babai's closest vertex algorithm

This function utilizes the possibilities of numPy as a linear algebra library. Here the set of linear Equations 1.13, as shown in the example are being solved using

[1]https://asecuritysite.com/encryption/lattice_plot

`linalg.solve()` function, which takes two input parameters. First is a set of coefficients of the unknowns (**b** matrix shown in Equation 3.1), and the second one is a vector of solutions to these equations, in the form of

$$\begin{pmatrix} b_{11}t_1 & b_{12}t_2 \\ b_{21}t_1 & b_{22}t_2 \end{pmatrix} = \begin{pmatrix} \mathbf{t}_0 \\ \mathbf{t}_1 \end{pmatrix}, \tag{3.1}$$

Next step, as shown in the example Equation 1.15 is to round the solutions, which is done using the `round()` function. Finally, the algorithm needs to multiply the generating matrix with the coefficients found in the previous step. This can be achieved by calculating the dot product of the two matrices, which is done using `dot()` function.

```
def solve_babai(basis, t):
    res = np.array([t[0],t[1]])
    a = np.round(np.linalg.solve(basis, res))
    return np.dot(a, basis)
```

Listing 3.6: Babai's algorithm function.

3.4.4 Generating LWE protocol

The function first generates the necessary parmeters according to the proposed boundaries. Then, the function utilizes the uniform random distribution `np.random.randint()` to populate the matrices and vectors. These are stored in variables `m, A, s, e`. Following the protocol, the LWE result is calculated. Next, the encryption parameters pair is calculated in variables `ap, pp`. The decryption happens in variable `ep`, and the whole function returns `decrypted` parameter based on the location of the nearest case. This step was produced by utilizing the ternary operator, which combines the condition and return value code in one line. Listing 3.7 depicts the aforementioned algorithm.

```
m = int(np.ceil(1.1 * np.log(q) * n))
A = np.random.randint(low=-q,high=q,size=(m,n))
s = np.random.randint(low=-q,high=q,size=n)
e = np.random.randint(-1,1,size=m)

s = np.transpose(s)
e = np.transpose(e)

B = np.mod(np.add(np.dot(A,s), e), q)
ap = np.mod(A.sum(axis=0),q)
```

```
pp = (B.sum() + message * q//2) % q

ep = (pp - np.dot(ap, s)) % q

decrypted = 1 if (q//2-(q//4)) < ep < (q//2+(q//4)) else
    0
```

Listing 3.7: Generating random LWE problem.

3.5 Transforming point coordinates

To plot an element of a multiplicative group on a unit circle, we need to solve a transformation problem from line coordinates to coordinates on a unit circle. This is done by a known transformation

$$x = \sin(\frac{2\pi e}{p}), y = \cos(\frac{2\pi e}{p}), \tag{3.2}$$

where p is the group modulus and e is the element. Equations 3.2 is implemented using numpy as depicted in Listing 3.8. These coordinates are later plotted as point with coresponding x and y coordinates.

```
epx=np.sin((2*np.pi/17)*ep)
epy=np.cos((2*np.pi/17)*ep)
```

Listing 3.8: Transformation of coordinates.

3.6 Plotting the graphs

As stated in Section 2.4.2, Bokeh uses figure objects to visualize data from data source.

For the purpose of storing data, the `ColumnDataSource` class was utilized, as it allows the plots to be subscribed to the values and be dynamically changed in real time. To change the values when an action is performed, callbacks were utilized, which are being executed by the `on_change()` function.

3.6.1 Data model

`ColumnDataSource` class is at the core of the model. Module `babai.py` contains three objects of this type. Note that the only limiting factor is the condition the columns have to have the same length, which was the reason behind creating three objects. Object source contains vectors in the lattice, bsource contains the basis, the unimodular matrix and hadamard ratios and csource contains coordinates of the selected vector, calculated vector and the linear independence boolean of the basis.

```
source = ColumnDataSource(data=dict(x=x, y=y))
```

Listing 3.9: Data model example.

3.6.2 Plotting and callbacks

The creation of a simple plot can be seen in Listing 2.2. One important property we have utilized is the ability of glyphs to subscribe to a source. When the source has been changed, the glyph will be automatically updated. Listing 3.10 shows how another glyph is being created and subscribed to the bsource object.

```
p2.add_layout(Arrow(end=OpenHead(line_color="firebrick",
    line_width=4), x_start=0, y_start=0, x_end='xu', y_end
    ='yu', source=bsource))
```

Listing 3.10: Plotting points and arrows.

We have shown how to update plots with the data objects. Now we need to figure out how to update the data objects. That is done using callbacks, which are update functions executed on a condition stated by the on_change() function. Listing 3.11 shows a TextInput object being initialized, which will be triggering the x1_callback function every time the value changes. This callbck will now create a new dictionary with the changed values and put it into the proper data object.

```
x1_input = TextInput(value="2", title="X1:")
x1_input.on_change('value', x1_callback)
```

Listing 3.11: Trigger based on change of an input.

3.7 Custom JavaScript callbacks

Even though one of the main advantages of Bokeh library is the fact that the developer does not have to write native JavaScript code due to the vast amounts of models available in the API, some advanced functionalities can be written and subsequently added to the working Python project. In this case, we have decided that visualizing larger matrices and their subsequent operations on them are too inefficient to properly visualize. Instead of limiting users to a low matrix dimensions, where the protocol is not intended to be used, we have decided that users should have the ability to download the data and use them for further inspection. For this reason, we have implemented the download functionality, which parses the `ColumnDataSource` objects into a JSON file and downloads it on a click of a button. Listing 3.12 depicts the parser function outputting JSON objects.

```
download_button.js_on_event("button_click", CustomJS(args=dict(source=lwe_to_json(bsource.data['A'], csource.data['s'], dsource.data['e'], bsource.data['B'], csource.data['ap'], source.data['pp'][0] ,source.data['ciph'][0] ,source.data['dec'][0])), code=open(join(dirname(__file__), "download.js")).read()))

def lwe_to_json(A, s, e, B, ap, pp ,ep ,dec):
    ret = {"A": A,
        "s": s,
        "e": e,
        "B": B,
        "ap": ap,
        "pp": int(pp),
        "ep": int(ep),
        "dec": int(dec)}

    enc = json.dumps(ret, cls=NumpyArrayEncoder)
    return enc
```

Listing 3.12: Generating random LWE problem.

This functionality is not natively supported by Bokeh, although the documentation provides steps on achieving this use case. The parser has to be implemented in JavaScript, as it's functioning on the client-side, where the web browser is not able to utilize Python. Listing 3.13 depicts the download function inspired by examples found in Bokeh repository. This function is triggered by the `download_button.js_on_event` callback, that parses the data from `ColumnDataSource` objects into JSON.

```
const filename = 'lwe_params.csv'
const filetext = source

const blob = new Blob([filetext], { type: 'text/json;charset=utf-8;' })

...
```

```
    const link = document.createElement('a')
    link.href = URL.createObjectURL(blob)
    link.download = filename
    link.target = '_blank'
    link.style.visibility = 'hidden'
    link.dispatchEvent(new MouseEvent('click'))
}
```

Listing 3.13: JavaScript download callback.

3.8 Rendering HTML

Bokeh natively supports creating Div objects that render HTML text. This object can also render LaTex code snippets using MathJax JavaScript library. For this use case we utilize the Python raw strings shown with `r` key before the start of the string as opposed to the formatted strings starting with `f` key. Raw strings render the text input as is, which means this string ignores all special characters, such as parentheses and slashes. For this reason, a parser function formatting numPy arrays into LaTex pmatrix matrices was developed. This parser is heavily inspired by code found on StackOverflow [2]. Listing 3.14 depicts a simple Div object that contains HTML tags, parser function and the object containing both raw and formatted Python strings.

```
key = Div(text="Key␣Generator", width=300, height=30)

def pmatrix(a):
    lines = str(a).replace('[', '').replace(']', '').splitlines()
    rv = [r'\begin{pmatrix}']
    rv += ['␣␣' + '␣&␣'.join(l.split()) + r'\\' for l in lines]
    rv +=  [r'\end{pmatrix}']
    return '\n'.join(rv)

enc_pair = Div(text=r"$$a'␣=␣\sum_I␣" + f"{pmatrix(bsource.data['A'])}␣=␣{pmatrix(csource.data['ap'])}$$", width=300, height=150)
```

Listing 3.14: LaTex parser.

[2]https://stackoverflow.com/questions/17129290/numpy-2d-and-1d-array-to-latex-bmatrix

3.9 Extending the application

The application was concieved with future expansion in mind. For this reason, developing new pages and modules is trivial. Listing 3.15 depicts the core structure of the Bokeh module located in the app folder. The module requires a router with render function added into views.py file as depicted in Listing 3.3. Furthermore the module needs to have an endpoint reference in the Server object in run.py file as depicted in Listing 3.17.

The module widgets and plots can be arranged by utilizing the row() and column() functions into a grid. Then the structure has to be added into the root of the document.

```
def <module name>(doc):
    ##Module source code
    module = row(widget, plot)
    doc.add_root(module)
    doc.title = "New␣module␣name"
```

Listing 3.15: Bokeh module template.

3.10 Docker

To allow the application to be easily deployed on any hardware, the application is wrapped in a Docker container, which is described in the Dockerfile. As shown below, this image is based on the official Python image developed by Docker. After downloading and setting up the image, it creates the app directory to which the whole repository is copied into. It is followed by installing the requirements as described in Section 3.2. Now the application ports are being exposed, but not published. This is achieved by using the `EXPOSE` keyword. The publishing will happen when creating the container by using the -p flag, which will map these ports to assigned external ports. At last, the application is being started with the command `python run.py`. Listing 3.16 depicts the Dockerfile used in the final application.

```
FROM python:3.9.7

WORKDIR /app
COPY . .
ENV IP=159.223.216.239
RUN pip install -r requirements.txt
EXPOSE 80:80
EXPOSE 50007:50007
ENTRYPOINT ["python"]
CMD ["run.py"]
```

Listing 3.16: Dockerfile.

3.11 Parallelism

Bokeh server is serving each client in a separate session. Furthermore, Bokeh server and Flask server are running in separate threads. This is achieved by using the `tornado` and `threading` libraries. These libraries provide the application with separatate IO loop and the thread respectively. Listing 3.17 depicts the `bk_worker` function containing the initialization of the server as well as the start of the IO loop.

```
def bk_worker():
    server = Server({'/babai' : babai_app}, io_loop=
       IOLoop(), allow_websocket_origin=["*"], port
       =50007)
    server.start()
    server.io_loop.start()

Thread(target=bk_worker).start()
```

Listing 3.17: Worker function.

3.12 Security

The possible attack vectors include a Denial of Service (DoS) attack targeted on the Bokeh server, where the attacker would create more sessions than the server could handle. There are several risks of arbitrary code execution, since the Bokeh library is not hardened by default. Possible mitigations include limiting sessions from which could be the Bokeh server sessions created.

Utilizing reverse proxy that handles DoS attacks protection, such as CloudFlare would mitigate the risk of DoS attacks. Since all of the input parameters are being sanitized, the risk of arbitrary code execution is also minimal. The Python modules accept requests made only from the sites the modules are located, which means that the modules cannot be accessed externally. This was achieved by utilizing the `allow_websocket_origin` parameter, which can be considered a security hardening measure.

3.13 User interface

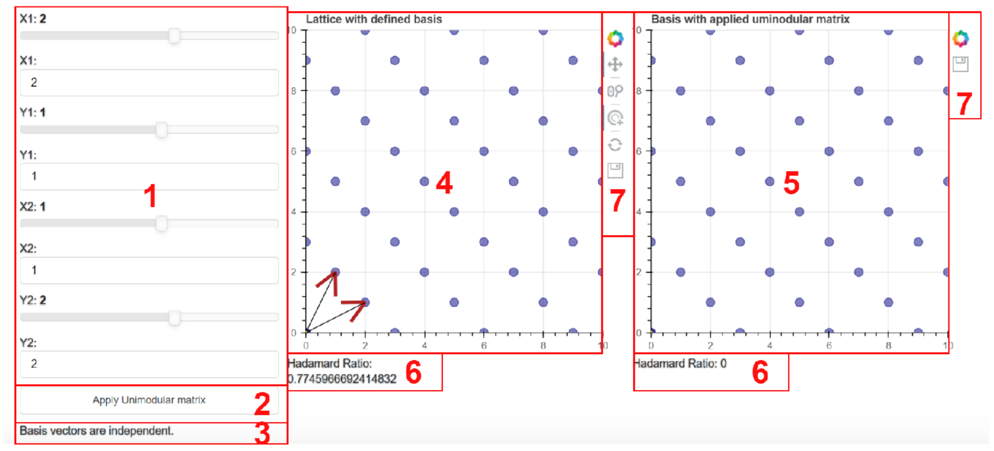
Web interface is a single page application, where users can access the Bokeh module and find out about the use of the application.

3.13.1 Babai module

Figure 3.3 shows the interface of the Babai module separated into functional blocks:

1. This section allows users to select the basis vectors. Users can use the sliders for real-time plotting of the lattice, or can type them directly into the boxes below.
2. Here the users will generate random unimodular matrix, which will be applied to the basis vectors selected above.
3. Indicator of linear independence.
4. In this plot users see the selected bases and generated portion of the lattice.
5. In this plot users see the same lattice, but with unimodular matrix applied to the bases.
6. Indicators of Hadamard ratio for each basis.
7. Bokeh plot toolbar (from top to bottom);
    - Pan (Used to move around the plots)
    - Zoom (Used to zoom to a current section of the plots)
    - Tap (Babai's closest vertex algorithm)
    - Reset (Resets the plots to the default view)
    - Save (Generates an image of the current view of the plot)

Note the toolbar is different on the second plot due to the controls being tied to both plots simultaneously, which leaves the second plot only with the save tool.


X1: 2

X1:
2

Y1: 1

Y1:
1

X2: 1

X2:
1

Y2: 2

Y2:
2

Apply Unimodular matrix

Basis vectors are independent.

Hadamard Ratio:
0.7745966692414832

Hadamard Ratio: 0

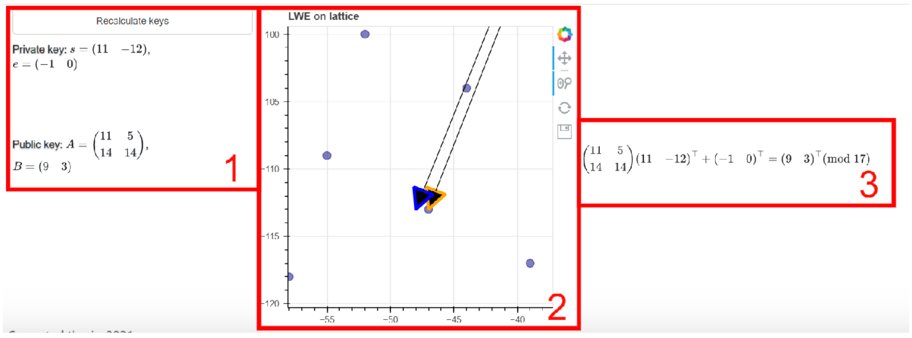
1
2
3
Lattice with defined basis
4
7
Basis with applied unimodular matrix
5
7
6
6

Fig. 3.3: User Interface of the Babai module.

3.13.2 Lattice/LWE module

Figure 3.4 shows the interface of the Lattice/LWE module separated into functional blocks:

1. This section allows users to randomize the lattice bases as well as outputs the generated keys.
2. In this plot users see the highlighted lattice bases as well as the .
3. This section allows users to select the message bit they want to encrypt. After the selection users click on Encrypt the message button.



Private key: $s = (11 \quad -12)$, $e = (-1 \quad 0)$

Public key: $A = \begin{pmatrix} 11 & 5 \\ 14 & 14 \end{pmatrix}$, $B = (9 \quad 3)$

$\begin{pmatrix} 11 & 5 \\ 14 & 14 \end{pmatrix}(11 \quad -12)^{\top} + (-1 \quad 0)^{\top} = (9 \quad 3)^{\top} \pmod{17}$

Fig. 3.4: User Interface of the Lattice/LWE module.

Workflow

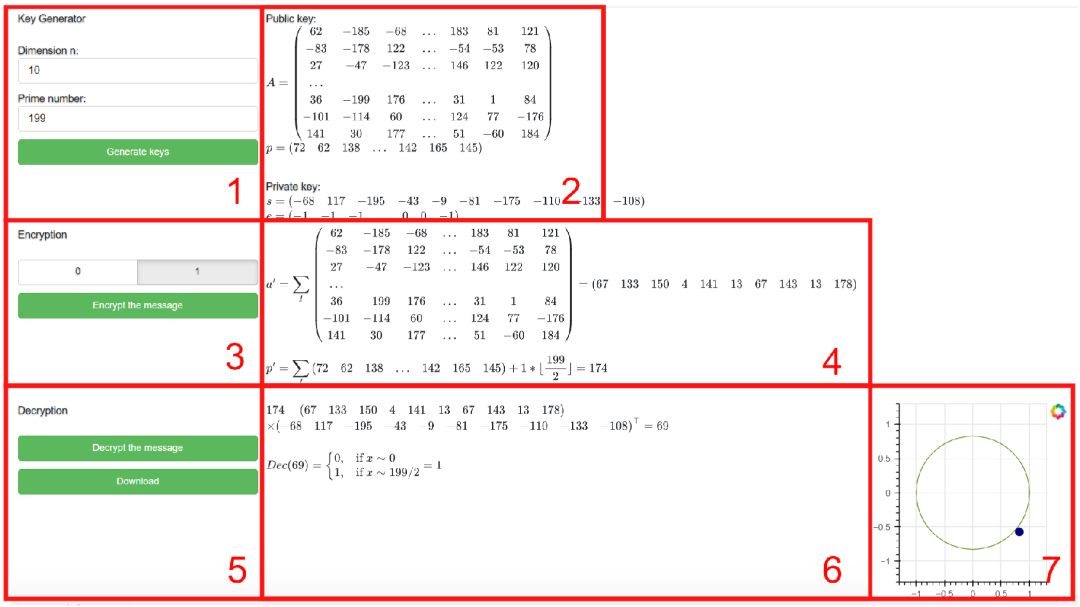
This module contains only one point of user interaction, the Recalculate keys button. Each click generates new basis, as well as new LWE problem. Users can see the generated parameters on the plot and that it corresponds to the Closest Vector Problem from the previous module. Users can also see the rendered LWE equation from which the parameters are taken.

3.13.3 LWE protocol module

Figure 3.5 shows the interface of the LWE protocol module separated into functional blocks:

1. This section allows users to input the matrix dimension and the prime modulus. After the selection is done, user clicks on Generate keys button.
2. Here the users see the generated public and private keys.
3. This section allows users to select the message bit they want to encrypt. After the selection users click on Encrypt the message button.
4. Here the users see the encrypted message parameters.

5. This section allows users to decrypt the message and download the parameters as JSON object.
6. Here the users see the decryption calcultions.
7. This plot depicts the decryption on a unit circle.



Dimension n: 10

Prime number: 199

Public key:

$$A = \begin{pmatrix} 62 & -185 & -68 & \dots & 183 & 81 & 121 \\ -83 & -178 & 122 & \dots & -54 & -53 & 78 \\ 27 & -47 & -123 & \dots & 146 & 122 & 120 \\ \dots \\ 36 & -199 & 176 & \dots & 31 & 1 & 84 \\ -101 & -114 & 60 & \dots & 124 & 77 & -176 \\ 141 & 30 & 177 & \dots & 51 & -60 & 184 \end{pmatrix}$$

$p = (72 \quad 62 \quad 138 \quad \dots \quad 142 \quad 165 \quad 145)$

Private key:

$s = (-68 \quad 117 \quad -195 \quad -43 \quad -9 \quad -81 \quad -175 \quad -110 \quad -133 \quad -108)$

$e = (-1 \quad -1 \quad -1 \quad \dots \quad 0 \quad 0 \quad -1)$

0 [✓ 1]

$$a' = \sum_{I} \begin{pmatrix} 62 & -185 & -68 & \dots & 183 & 81 & 121 \\ -83 & -178 & 122 & \dots & -54 & -53 & 78 \\ 27 & -47 & -123 & \dots & 146 & 122 & 120 \\ \dots \\ 36 & 199 & 176 & \dots & 31 & 1 & 84 \\ -101 & -114 & 60 & \dots & 124 & 77 & -176 \\ 141 & 30 & 177 & \dots & 51 & -60 & 184 \end{pmatrix} = (67 \quad 133 \quad 150 \quad 4 \quad 141 \quad 13 \quad 67 \quad 143 \quad 13 \quad 178)$$

$$p' = \sum_{I} (72 \quad 62 \quad 138 \quad \dots \quad 142 \quad 165 \quad 145) + 1 * \lfloor \frac{199}{2} \rfloor = 174$$

$$174 \quad (67 \quad 133 \quad 150 \quad 4 \quad 141 \quad 13 \quad 67 \quad 143 \quad 13 \quad 178) \times (-68 \quad 117 \quad -195 \quad -43 \quad -9 \quad -81 \quad -175 \quad -110 \quad -133 \quad -108)^{\top} = 69$$

$$Dec(69) = \begin{cases} 0, & \text{if } x \sim 0 \\ 1, & \text{if } x \sim 199/2 \end{cases} = 1$$

Fig. 3.5: User Interface of the LWE protocol module.

Workflow

This module has comparably simple user interface as the previous modules. First, users select the input parameters, after which they are able to generate keys. After the keys have been generated, users are able to select the bit message and encrypt the message using the pre-calculated keys. Then the users can decrypt the message by clicking the decryption button. Throughout the whole computations users see the rendered equations next to the buttons allocated for the given portion of the protocol. After decrypting the message, users can see that the decrypted parameter has been close either to the uppermost or lowermost point on the unit circle. After the protocol has been completed, the users can download the parameters in a JSON format using the Download button.

3.14 Installation

The installation consists of building the docker image and consequently running a container based on that image. This section presumes that user has docker already installed. Listing 3.18 shows the commands necessary for building and running the application. Note that in command `docker build`, the dot refers to a current directory. This means if the user is not in the application root directory, the user has to specify this path. To run the server in the background, user can specify the `-d` flag in the `docker run` command. While running in background, the container can be stopped using `docker stop lattice-server` command. Now the application is available on the local IP address on port 80.

```
docker build -t lattice .
docker run -p 80:80 -p 50007:50007 --name=lattice-server
   -t -i lattice
```

Listing 3.18: Building the image.

3.15 Deployment to DigitalOcean

DigitalOcean [24] is paid cloud hosting service designed to among others, easily host, deploy and monitor cloud applications. Since the application was developed as a Docker container, the deployment was possible in minutes. After selecting the node parameters, such as number of processors, RAM and storage, the user obtains ssh access to a Ubuntu instance with pre installed Docker. Then, the deployment is a matter of cloning the repository, building the image and running the container as described in Section 3.14.

Conclusion

This thesis was concerned with creating a user friendly application demonstrating key concepts in lattice-based cryptography. We studied the theoretical aspects of the thesis and presented the mathematical theory of lattices as well as some important properties of these structures. Moreover, we graphically demonstrated the Learning with errors problem and shown the Boyen encryption protocol.

Following the theory we have laid out the reasoning behind selection of the technology stack as well as technology used in the development process. Python programming language was chosen as the most suitable platform for it's extensive list of libraries that have aided us in the development. Among those are the libraries which allow developers extensive work with linear algebra, web server and real time plotting. We have laid out a case for using versioning system in the development process. We have shown how we have implemented the theoretical structures and the operations on them using numPy. Furthermore, we have shown what are the underlying server-side aspects of the application, such as the router function and Bokeh server, which is handling the session data in separate threads running on the server. We have also shown the development of custom JavaScript callbacks and security hardening. At last, we have described the process of deploying the server on public cloud.

The development concluded with a structure purposefully created as a foundation for easy development of new modules, which means the whole project can be used as a library for local computations or to improve on the original project as is the custom in the open-source community.

Bibliography

[1] *NSA paid $10 million to put its backdoor in RSA encryption, according to Reuters report* [online]. The Verge, 2013 [cit. 2021-12-09]. Available at: https://www.theverge.com/2013/12/20/5231006/nsa-paid-10-million-for-a-back-door-into-rsa-encryption-according-to

[2] WANG, Xiaoyun and Hongbo YU. How to Break MD5 and Other Hash Functions. In: *Advances in Cryptology — EUROCRYPT 2005* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, s. 19-35 [cit. 2021-12-09]. ISBN 9783540259107. ISSN 0302-9743. Available at: doi:10.1007/11426639_2

[3] BERNSTEIN, Daniel J., Johannes BUCHMANN and Erik DAHMEN. *Post-Quantum Cryptography.* Berlin, Heidelberg: Springer, 2009. ISBN 9783540887010. Available at: doi:10.1007/978-3-540-88702-7

[4] Post-Quantum Cryptography. *COMPUTER SECURITY RESOURCE CENTER* [online]. Gaithersburg, MD: National Institute of Standards and Technology, 2017 [cit. 2021-12-08]. Available at: https://csrc.nist.gov/projects/post-quantum-cryptography

[5] CHOW, Jerry, Oliver DIAL and Jay GAMBETTA. *IBM Quantum breaks the 100-qubit processor barrier* [online]. IBM, 2021 [cit. 2022-05-30]. Available at: https://research.ibm.com/blog/127-qubit-quantum-processor-eagle

[6] *Tanja Lange leads multi-million Euro project to protect data against quantum computers* [online]. Netherlands: Eindhoven University of Technology, 2015 [cit. 2022-05-30]. Available at: https://www.tue.nl/en/news/news-overview/23-04-2015-tanja-lange-leads-multi-million-euro-project-to-protect-data-against-quantum-computers/

[7] HOFFSTEIN, Jeff, Nick HOWGRAVE-GRAHAM, Jill PIPHER and William WHYTE. Practical Lattice-Based Cryptography: NTRUEncrypt and NTRUSign. *The LLL Algorithm.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, s. 349-390. ISBN 9783642022944. ISSN 1619-7100. Available at: doi:10.1007/978-3-642-02295-1_11

[8] DUCAS, L, T LEPOINT, V LYUBASHEVSKY, P SCHWABE, G SEILER and D STEHLE. CRYSTALS — Dilithium: Digital Signatures from Module Lattices. *IACR Transactions on Symmetric Cryptology* [online]. 2018, **2018**, 238-268 [cit. 2022-05-30]. ISSN 2519-173X.

[9] D-ANVERS, Jan-pieter, Angshuman KARMAKAR, Sujoy SINHA ROY and Frederik VERCAUTEREN. Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. 10831. Cham: Springer International Publishing, 2018, s. 282-305. ISBN 3319893386. ISSN 0302-9743. Available at: doi:10.1007/978-3-319-89339-6_16

[10] PRADHAN, Pawan Kumar, Sayan RAKSHIT and Sujoy DATTA. Lattice Based Cryptography: Its Applications, Areas of Interest & Future Scope. In: *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)* [online]. IEEE, 2019, 2019, s. 988-993 [cit. 2021-12-08]. ISBN 978-1-5386-7808-4. Available at: doi:10.1109/ICCMC.2019.8819706

[11] AGGARWAL, Divesh, Gavin BRENNEN, Troy LEE, Miklos SANTHA and Marco TOMAMICHEL. Quantum Attacks on Bitcoin, and How to Protect Against Them. *Ledger* [online]. 2018, **3** [cit. 2021-12-09]. ISSN 2379-5980. Available at: doi:10.5195/ledger.2018.127

[12] CHEN, Lily, Stephen JORDAN, Yi-Kai LIU, Dustin MOODY, Rene PERALTA, Ray PERLNER and Daniel SMITH-TONE. Report on Post-Quantum Cryptography. *National Institute of Standards and Technology: Internal Report 8105* [online]. US Department of Commerce, 2016, (12), 15 [cit. 2021-11-07]. Available at: https://nvlpubs.nist.gov/nistpubs/ir/2016/nist.ir.8105.pdf

[13] HOFFSTEIN, Jeffrey, Jill PIPHER and J. H SILVERMAN. *An Introduction to Mathematical Cryptography*. New York, NY: Springer New York, 2008. ISBN 9780387779935.

[14] VAN EMDE BOAS, Peter. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. *Report. Department of Mathematics. University of Amsterdam*. Department, Univ., 1981, **1981**(84), 10. Available at: https://staff.fnwi.uva.nl/p.vanemdeboas/vectors/abstract.html

[15] ROSS, Sheldon M. *A first course in probability*. 7th ed. Upper Saddle River: Pearson Prentice Hall, 2006, x, 565 s. : il. ISBN 0-13-185662-6.

[16] REGEV, Oded. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM* [online]. NEW YORK: ACM, 2009, **56**(6), 1-40 [cit. 2022-05-27]. ISSN 0004-5411. Available at: doi:10.1145/1568318.1568324

[17] BOYEN, Xavier. Expressive encryption systems from lattices (abstract from the invited lecture). In: *Lecture Notes in Computer Science (including subseries*

Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) [online]. 2011, s. 1-12 [cit. 2022-05-27]. ISBN 9783642255120. ISSN 0302-9743. Available at: doi:10.1007/978-3-642-25513-7_1

[18] CHEN, Zhigang, Jian WANG, Liqun CHEN and Xinxia SONG. A Regev-Type Fully Homomorphic Encryption Scheme Using Modulus Switching. *TheScientificWorld* [online]. LONDON: Hindawi Publishing Corporation, 2014, **2014**, 983862-12 [cit. 2022-05-27]. ISSN 2356-6140. Available at: doi:10.1155/2014/983862

[19] VAN ROSSUM, Guido. Python Tutorial. *Python Documentation* [online]. PythonLabs, 2001 [cit. 2021-12-08]. Available at: https://docs.python.org/2.0/tut/tut.html

[20] HARRIS, Charles R, K Jarrod MILLMAN, Stéfan J VAN DER WALT, et al. Array programming with NumPy. *Nature (London)* [online]. England: Nature Publishing Group, 2020, **585**(7825), 357-362 [cit. 2021-12-06]. ISSN 0028-0836. Available at: doi:10.1038/s41586-020-2649-2

[21] *Flask documentation* [online]. [cit. 2021-11-29]. Available at: https://flask.palletsprojects.com/en/2.0.x/

[22] *Bokeh documentation* [online]. 2021 [cit. 2021-11-29]. Available at: https://docs.bokeh.org/en/latest/index.html

[23] Containerization. *IBM Cloud Learn Hub* [online]. IBM, 2021 [cit. 2021-12-08]. Available at: https://www.ibm.com/cloud/learn/containerization

[24] *DigitalOcean documentation* [online]. [cit. 2022-05-27]. Available at: https://docs.digitalocean.com/

[25] *Jinja Documentation* [online]. Pallets, 2007 [cit. 2022-05-27]. Available at: https://jinja.palletsprojects.com/en/3.1.x/

A Installation Manual

After logging into the desired server or desktop, the first step is to clone the repository.

```
git clone https://github.com/xsecka04/crypto_lattice
cd crypto_lattice
```

Listing A.1: Cloning the Git repository.

A.1 Deployment with Docker

If the user has selected the deployment using Docker, user continues with building the Docker image and creating a container. Before building the container, the user needs to specify the domain from which the application will be hosted. In case of deployment on a server, user has to specify the environmental variable IP to be the external IP address of the server. In the case of local deployment, user can set the IP variable to localhost. The variable is located in the Dockerfile. When the variable is specified, user can proceed with building the image and running the container.

```
docker build -t lattice .
docker run -p 80:80 -p 50007:50007 --name=lattice-server
   -t -i lattice
```

Listing A.2: Building the image.

To stop the server from running, user can use the command

```
docker stop lattice-server
```

Listing A.3: Stopping the server from running.

If the container already exists and it is not running, user can start the container by using command

```
docker start lattice-server
```

Listing A.4: Starting the server.

A.2 Deployment with virtualenv

Although not recommended, users can locally deploy the application using Python virtual environment. This case is recommended only for development purposes. The user needs to have installed latest versions of Python and pip. Note that user needs to specify the environmental variable IP per the operating system the computer is running. The commands are as depicted in Listing A.5.

```
export IP=localhost #Linux
set IP=localhost #Windows
```

Listing A.5: Setting the environmental variable

If the user satisfies the aforementioned prerequisites, they can proceed to installing the virtulenv package and creating and activating the virtual environment.

```
pip install virtualenv
virtualenv venv
source venv/bin/activate
```

Listing A.6: Installation of virtualenv environment.

Then the user needs to install all of the required libraries and run the application.

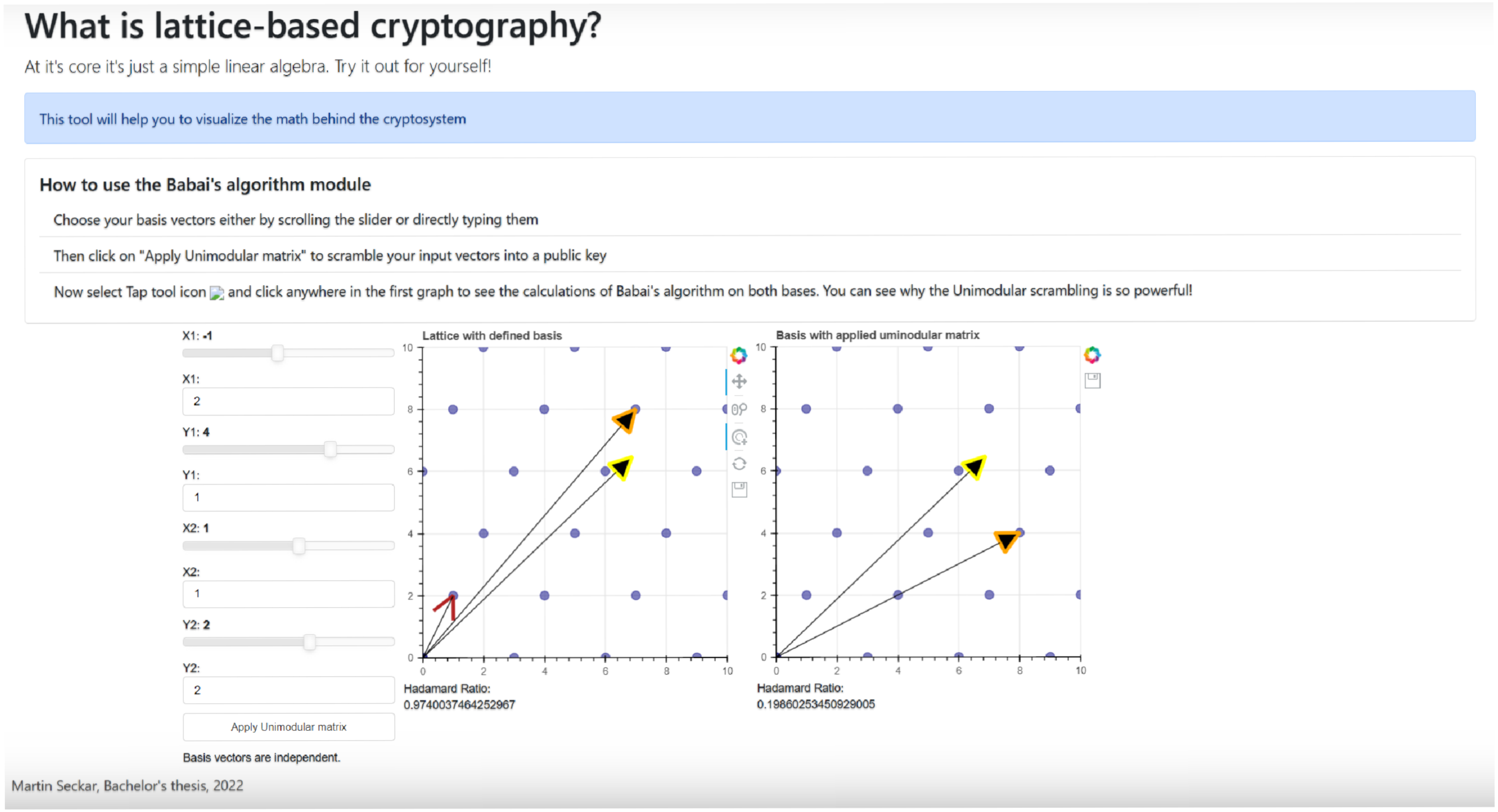
```
pip install -r requirements.txt
python run.py
```

Listing A.7: Running the application.

B User Manual

The application contains three modules with accompanying user instructions on the top of the module. These instructions are depicted on Figures B.1, B.2 and B.3.

The user is encouraged to start with the Babai module to understand the basics of lattices, bases and CVP. Then the user can see the similarities with the LWE problem in the CVP/LWE module. At last, the user can try to use the LWE protocol module where the user can experiment with the initial parameters.



What is lattice-based cryptography?

At it's core it's just a simple linear algebra. Try it out for yourself!

This tool will help you to visualize the math behind the cryptosystem

How to use the Babai's algorithm module

Choose your basis vectors either by scrolling the slider or directly typing them

Then click on "Apply Unimodular matrix" to scramble your input vectors into a public key

Now select Tap tool icon and click anywhere in the first graph to see the calculations of Babai's algorithm on both bases. You can see why the Unimodular scrambling is so powerful!

X1: -1

X1: 2

Y1: 4

Y1: 1

X2: 1

X2: 1

Y2: 2

Y2: 2

Apply Unimodular matrix

Basis vectors are independent.

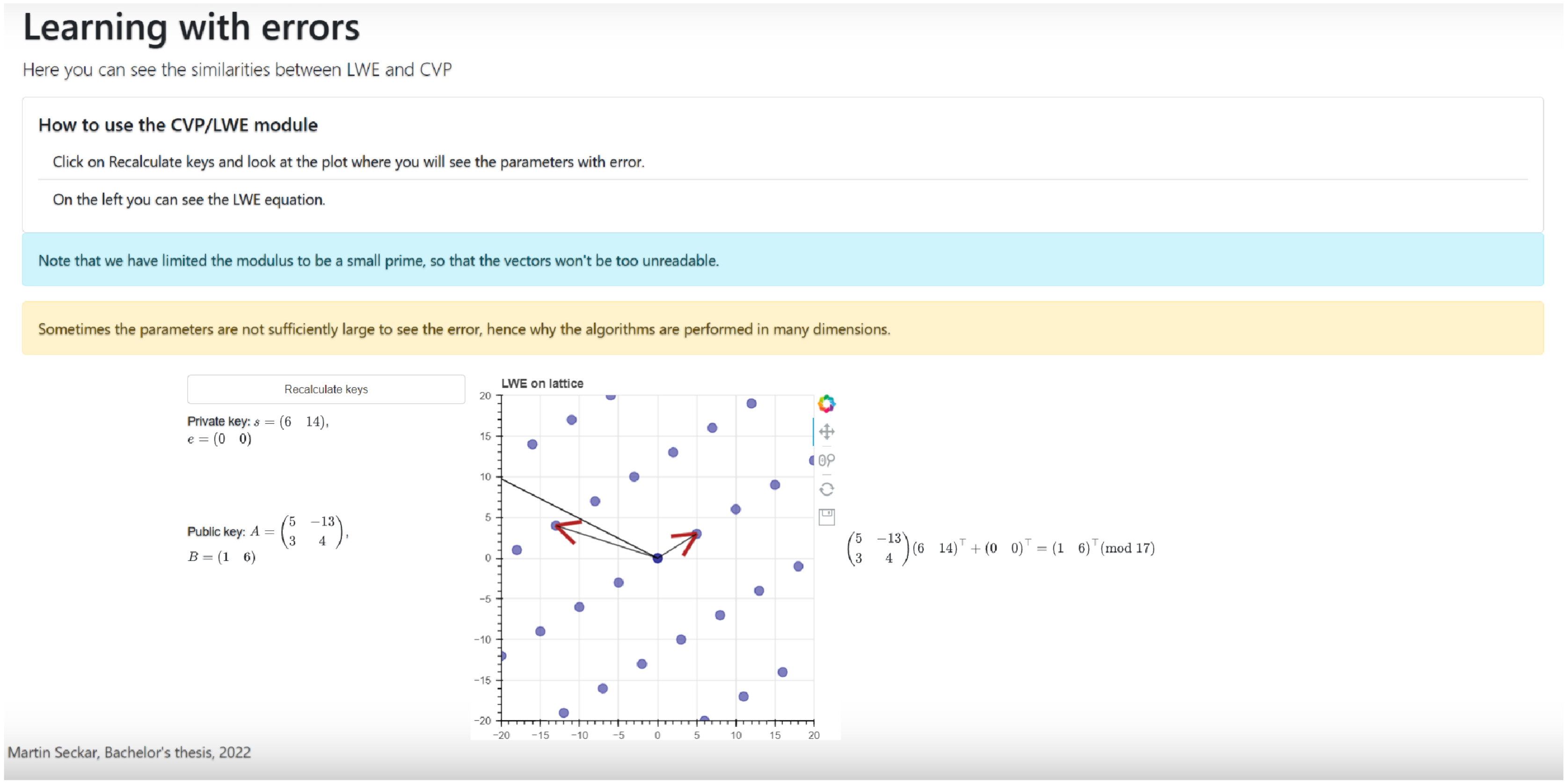
Hadamard Ratio: 0.9740037464252967

Hadamard Ratio: 0.19860253450929005

Martin Seckar, Bachelor's thesis, 2022

Lattice with defined basis
Basis with applied uminodular matrix
0
2
4
6
8
10

Fig. B.1: User instructions of the Babai module.



Learning with errors

Here you can see the similarities between LWE and CVP

How to use the CVP/LWE module

Click on Recalculate keys and look at the plot where you will see the parameters with error.

On the left you can see the LWE equation.

Note that we have limited the modulus to be a small prime, so that the vectors won't be too unreadable.

Sometimes the parameters are not sufficiently large to see the error, hence why the algorithms are performed in many dimensions.

Recalculate keys

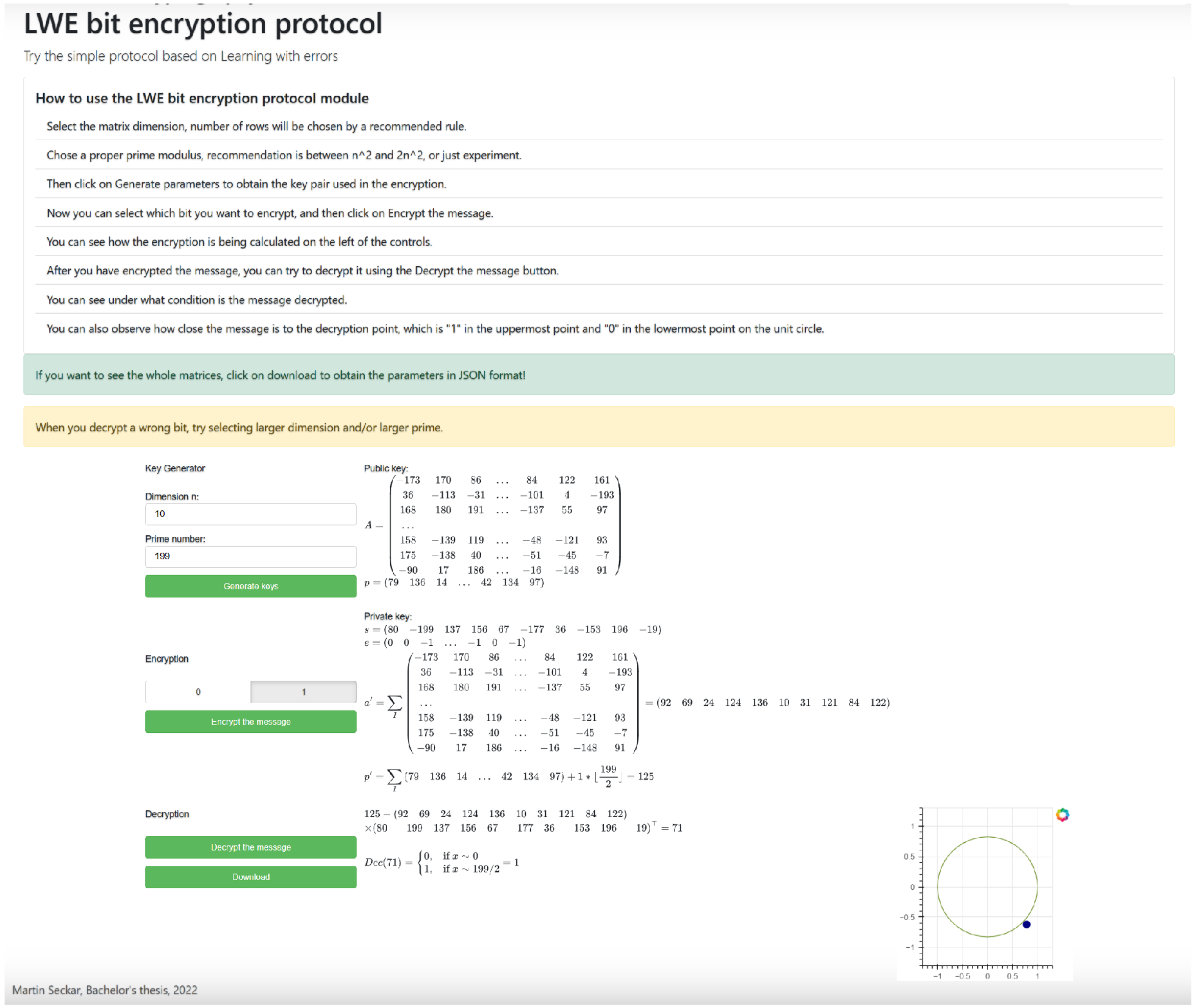
Private key: $s = (6 \quad 14)$,
$e = (0 \quad 0)$

Public key: $A = \begin{pmatrix} 5 & -13 \\ 3 & 4 \end{pmatrix}$,
$B = (1 \quad 6)$

$$\begin{pmatrix} 5 & -13 \\ 3 & 4 \end{pmatrix}(6 \quad 14)^{\top} + (0 \quad 0)^{\top} = (1 \quad 6)^{\top} \pmod{17}$$

Martin Seckar, Bachelor's thesis, 2022

Fig. B.2: User instructions of the CVP/LWE module.



LWE bit encryption protocol

Try the simple protocol based on Learning with errors

How to use the LWE bit encryption protocol module

Select the matrix dimension, number of rows will be chosen by a recommended rule.

Chose a proper prime modulus, recommendation is between n^2 and 2n^2, or just experiment.

Then click on Generate parameters to obtain the key pair used in the encryption.

Now you can select which bit you want to encrypt, and then click on Encrypt the message.

You can see how the encryption is being calculated on the left of the controls.

After you have encrypted the message, you can try to decrypt it using the Decrypt the message button.

You can see under what condition is the message decrypted.

You can also observe how close the message is to the decryption point, which is "1" in the uppermost point and "0" in the lowermost point on the unit circle.

If you want to see the whole matrices, click on download to obtain the parameters in JSON format!

When you decrypt a wrong bit, try selecting larger dimension and/or larger prime.

Key Generator

Dimension n: 10

Prime number: 199

Public key:

$$A = \begin{pmatrix} -173 & 170 & 86 & \dots & 84 & 122 & 161 \\ 36 & -113 & -31 & \dots & -101 & 4 & -193 \\ 168 & 180 & 191 & \dots & -137 & 55 & 97 \\ \dots \\ 158 & -139 & 119 & \dots & -48 & -121 & 93 \\ 175 & -138 & 40 & \dots & -51 & -45 & -7 \\ -90 & 17 & 186 & \dots & -16 & -148 & 91 \end{pmatrix}$$

$$p = (79 \quad 136 \quad 14 \quad \dots \quad 42 \quad 134 \quad 97)$$

Private key:

$$s = (80 \quad -199 \quad 137 \quad 156 \quad 67 \quad -177 \quad 36 \quad -153 \quad 196 \quad -19)$$

$$e = (0 \quad 0 \quad -1 \quad \dots \quad -1 \quad 0 \quad -1)$$

Encryption

0 1

$$a' = \sum_I \begin{pmatrix} -173 & 170 & 86 & \dots & 84 & 122 & 161 \\ 36 & -113 & -31 & \dots & -101 & 4 & -193 \\ 168 & 180 & 191 & \dots & -137 & 55 & 97 \\ \dots \\ 158 & -139 & 119 & \dots & -48 & -121 & 93 \\ 175 & -138 & 40 & \dots & -51 & -45 & -7 \\ -90 & 17 & 186 & \dots & -16 & -148 & 91 \end{pmatrix} = (92 \quad 69 \quad 24 \quad 124 \quad 136 \quad 10 \quad 31 \quad 121 \quad 84 \quad 122)$$

$$p' = \sum_I (79 \quad 136 \quad 14 \quad \dots \quad 42 \quad 134 \quad 97) + 1 * \lfloor \frac{199}{2} \rfloor = 125$$

Decryption

$$125 - (92 \quad 69 \quad 24 \quad 124 \quad 136 \quad 10 \quad 31 \quad 121 \quad 84 \quad 122) \times (80 \quad 199 \quad 137 \quad 156 \quad 67 \quad 177 \quad 36 \quad 153 \quad 196 \quad 19)^{\top} = 71$$

$$Dec(71) = \begin{cases} 0, & \text{if } x \sim 0 \\ 1, & \text{if } x \sim 199/2 \end{cases} = 1$$

Martin Seckar, Bachelor's thesis, 2022

Fig. B.3: User instructions of the LWE protocol module.