Utrecht University

Master Thesis
Faculty of Science
Department of Mathematics

Cryptanalysis of lattice-based Cryptography.

The effect of sparse ternary secrets on the hardness of Learning with Errors.

Author
P.A. van der Sluis

Daily Supervisor
S.H.S de Vries, MSc

First Reader
Prof. dr. C.F. Faber

Second Reader
Prof. dr. G.L.M. Cornelissen

Date
November 12, 2020

Contents

1 Introduction.

1.1 Motivation.

Cryptography is the study of secure communication over insecure channels in the presence of adversaries. The ancient Romans already had methods for encrypting their messages for safe communication. Julius Ceasar used a cipher for military messages which encoded messages by a rotational alphabetic shift of a fixed number (encoding 'a' to 'b', 'b' to 'c', etc.). At this point in history, this was already more than enough security as at that moment most people weren't able to read at all. Nowadays, cryptographic schemes are much more involved and harder to break. We roughly distinguish modern cryptographic systems into two categories: symmetric and asymmetric cryptography.

First, In symmetric cryptographic systems the encryption and decryption is done with the same key, possibly with a small transformation between the two keys. In order to safely communicate with someone, a parties need to set up a shared secret with the party they wish to communicate with.

Second, Asymmetric- or public-key- cryptography makes use of a pair of keys that don't have an easy transformation between them, namely: a public key and a private key. As the name may suggest, the public key is available to everybody interested, including the adversary. On the other hand, The private key is only known to the owner. Any person can encrypt any message using the public key, and can send it to the private key holder. With the private key the message can be decrypted. It is very important that an adversary cannot recover the private key from the public information available, and cannot decrypt a ciphertext without knowlegde about the private key.

Both types of cryptography have their own advantages and disadvantages. In general, symmetric cryptographic communication can be calculated more efficiently, making it more interesting when lots of data have to be encrypted. Asymmetric cryptography (given that the system is well-designed) has the advantage that for secure communication we only need to ensure that the private key is kept private. This can be achieved easily, as we only need the private key in one place and it doesn't need to be transmitted in any way. This makes key management very easy. The public key, only used to encrypt messages, is safe for anyone to have. The main disadvantage of asymmetric cryptography is that it is relative slow compared to symmetric cryptography.

In practice, one could combine these two forms of cryptography to benefit from the advantages of both types, which is called a Key Encapsulation Mechanism (KEM). Two parties can agree on a shared secret using asymmetric cryptography, only requiring the integrity of the public key. With this shared secret they can encrypt the larger data using the faster symmetric cryptography. In this way we have established a way to encrypt fast, with easy key management.

The safety of most currently used cryptosystems relies on the fact that there is no currently known efficient algorithm to solve a certain problem. Examples of these kind of problems are the factoring problem (split a composite number n into its prime factors) or the discrete logarithm problem (solve $a^b \equiv c \mod p$ for b). However, in 1994 Peter Shor came up with an algorithm [50] that runs on a quantum computer, which can break many of these cryptosystems. The value of this algorithm was at this point in time purely theoretical, as a working large scale quantum computer did not exist. This however, increased the importance of developing quantum computers. As time passed by, and the development of quantum computers did not stop, the urge for cryptosystems that are secure under the attack of a quantum computer became large. Even at this point in time, there is still no publicly known working quantum computer

available that has sufficient processing power to be a serious threat to modern cryptography. Hence the cryptographic systems that we currently use are still considered safe. However, it is not unreasonable to assume that in a couple of decades there is a working large scale quantum computer. As a consequence, an adversary collecting encrypted messages that contain information that still has to be secret even after decades, can all of a sudden decrypt these messages using this quantum computer. Because of that, there is urge to develop cryptographic systems that are safe under the attack of a quantum computer.

In 2017, based on this quantum threat, the National Institute of Standards and Technology (NIST) started a post quantum cryptography standardization process [41]. This is an open contest, where cryptographers can submit their proposed quantum resistant encryption methods. At this point in time, several post quantum cryptosystems were proposed. The aim of the standardization process was to do further research into these proposals to increase confidence in their security which should ultimately lead to new cryptographic standards.

The security of proposed cryptosystems for the NIST competition relies on the hardness of several different problems. The most important categories are: Lattice based Cryptography, Multivariate Cryptography, Code based Cryptography and Supersingular Elliptic Curve isogeny Cryptography. In this paper I study the Learning With Errors (LWE) problem and cryptosystems that are based on this problem, which is the most common problem on which the candidates are based. This is a form of Lattice based Cryptography.

In this thesis I first present an overview of the available literature. Next, two of the best known attacks, making use of lattice techniques are introduced and applied. After that, I consider some variants of the problem that are considered optimizations for cryptographic purposes and how they affect the security of these systems. Some examples of the cryptographic improvements these optimizations acquire are faster encryption and decryption, reduced public key sizes or ciphertext sizes and reduced memory usage. Most of the thesis will be used to describe the effect of sparse secretes. The NIST candidates Lizard [22] and nRound2 [12] are examples that uses this optimization. Furthermore, the effect on the security of the choice of polynomial modulus is discussed. The reader of this thesis should keep in mind that the goal of the thesis is to discuss these optimizations and to describe the tradeoff between security and efficiency.

1.2 Outline.

This thesis started with this section, where I described the urgency of the problem and put it in context. In section 2 common notation and terminology is listed that is used througout this thesis, that might differ slightly from other literature. Next, I start introducing the LWE problem in section 3. Based on a cryptosystem that can be reduced to the LWE problem, practical applications of the LWE problem are introduced. In this section I will also define some different forms of the LWE problem that occur in practice and how they correlate. As it turns out, the best currently known attacks against the LWE problem use methods to reduce the problem to a lattice problem. In section 4, I introduce lattices and generate a toolbox to perform lattice based attacks on the LWE problem. With this toolbox we use section 5 to describe two important attacks that are considered to be the most competitive known attacks, the primal and dual attack. In this section I also discuss conditions that are imposed on the attack parameters in order to be successful and test the heuristics that are used to determine these parameters. Then in section 6 I discuss some optimizations of the LWE problem that make the LWE problem more suitable for cryptographic purposes. The main optimization that is discussed is the use of secrets with low Hamming weight (i.e., the number of nonzero entries of the secret). Aditionally, I will also introduce a hybrid attack that exploits this low Hamming weight, and might outperform the primal and dual attack. In section 7 I further analyze this attack and answer the question how low a designer of a parameter set can go in Hamming

weight without losing too much security. Finally, In section 8 I discuss the value of my work and place the results in context.

1.3 My contributions

In this thesis I try to introduce the reader to as much as possible different forms of and attacks against the LWE problem. I have attempted to explain the existing literature as easily as possible to a reader unfamiliar with the problem or applications. First, In sections 3-5 I have summarized and combined some important literature on the LWE problem, the relevant literatature on lattices and two of the most important attacks against the LWE problem. In these sections I tried to make the literature intuitively understandable for the reader with small scale examples and justifications of the given heuristics. Second, In section 5 I also applied the known attacks to LWE problems with parameters chosen in such a way that the attacks are still feasible for a simple computer. At the end I also applied my toolbox to make estimates on the cost of an attack against the parameters of NIST candidates and compared them against the security claims of the authors of the proposed cryptosystems. Finally, in sections 6 and 7 I introduce and discuss an LWE optimization: sparse secrets. After reading these sections, the reader should have some feeling on how these sparse secrets affect the general security of the system. The aim of these sections is to show that the reader can properly pick a sparse enough Hamming weight, without losing too much security.

2 Terminology and Notation.

The following terminology and notation is used throughout this whole paper:

$\mathbb{Z}_q^n$	Set of n-dimensional vectors with integer coefficients modulo q.
n	Dimension of the LWE problem.
m	Number of Samples.
q	Modulus.
h	Hamming weight of a vector, i.e. number of nonzero entries.
σ	Standard deviation.
β	Block size of a BKZ algorithm.
δ_0	Hermite factor.
k	Hybrid dimension.
$\log(x)$	Logarithms are base 2, unless clearly indicated otherwise.
χ	Some probability distribution centered around 0.
$e \leftarrow_\$ \chi$	e is a random sample from the probability distribution χ.
$\mathbf{A} \leftarrow$ `Func()`	Assignment of the result of the function `Func()` to $\mathbf{A}$.
$\mathbf{A}$	Matrices denoted by capital boldface letters.
$\mathbf{a_i}$	i-th column of $\mathbf{A}$.
a_{ij}	Entry of $\mathbf{A}$ in the i-th row and j-th column.
$\mathbf{a}$	Vectors denoted by boldface lowercase letters.
a_i	i-th entry of $\mathbf{a}$.
$\mathbf{a} \cdot \mathbf{b}$	Inner product between $\mathbf{a}$ and $\mathbf{b}$: $\mathbf{a}^T\mathbf{b}$.
$\|\|\mathbf{a}\|\|$	Length of the vector $\mathbf{a}$: $\sqrt{\mathbf{a} \cdot \mathbf{a}}$.
$\mathbf{A}^T$	Transponentiation of matrix $\mathbf{A}$.
$\mathbf{I}_n$	n-dimensional identity matrix.
$\mathbf{0}_{m \times n}$	m times n all zero matrix. In case the context makes the dimension obvious, the subscript is omitted.
$(\mathbf{A}\|\|\mathbf{B})$	The matrix obtained by appending $\mathbf{B}$ to $\mathbf{A}$. With slight abuse of notation we will use this also for appending two column vectors underneath each other.
$\lceil x \rfloor$	Rounding a real number $x \in \mathbb{R}$ to the nearest integer.
$\mathcal{B}(\mathbf{x}, R)$	A ball in $\mathbb{R}^n$ centered at $\mathbf{x}$ of radius R.
$\mathcal{U}(S)$	Uniform probability distribution over a finite set S.

3 Learning with Errors.

3.1 LWE based Cryptosystems.

To introduce the Learning with Errors (LWE) problem and its practical relevance, I start by describing a supposedly quantum secure encryption scheme. As it turns out later in this section, one is able to reduce the problem of breaking this cryptosystem to solving the Learning with Errors problem. This problem is better described in the publicly available literature. I picked one of the alternative round 3 submissions for the NIST post-quantum standardization process [41], FrodoKEM. This is a key encapsulation mechanism that relies on the hardness of one of most basic forms of the LWE-problem. A brief simplified version of encryption and decryption is described below.

The cryptosystem uses two functions encode $ec(\cdot)$ and decode $dc(\cdot)$ that encode and decode a message k into a integer matrix (with some modulus q) such that $dc(ec(k)+\mathbf{N}) = k$ for a small enough integer error matrix $\mathbf{N}$. The system uses matrices that have dimensions $n, \bar{n}, \bar{m}$, where $\bar{n}, \bar{m} \equiv 0 \mod 8$. Alice constructs a private key, a matrix $\mathbf{S} \in \mathbb{Z}_q^{n\times\bar{n}}$. Also she constructs the public key consisting of two matrices, $(\mathbf{A}, \mathbf{B} = \mathbf{AS} + \mathbf{E}) \in \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^{n\times\bar{n}}$, where $\mathbf{E} \in \mathbb{Z}_q^{n\times\bar{n}}$ is a small random matrix.

Algorithm 1: Frodo.KeyGen()

Input: None;
$\mathbf{A} \leftarrow_{\$}$ `generateUniformMatrix()`;
$\mathbf{S}, \mathbf{E} \leftarrow_{\$}$ `generateSmallMatrix()`;
$\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$;
Output: Public key $pk \leftarrow (\mathbf{A}, \mathbf{B})$, Secret key $sk \leftarrow \mathbf{S}$;

Bob encrypts a message μ by creating three small matrices $\mathbf{S}', \mathbf{E}' \in \mathbb{Z}_q^{\bar{m}\times n}, \mathbf{E}'' \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$ and computing $\mathbf{B}' = \mathbf{S}'\mathbf{A} + \mathbf{E}'$ and $\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}''$. The ciphertext he sends to Alice is then the pair $(\mathbf{C}_1, \mathbf{C}_2) = (\mathbf{B}', \mathbf{V} + ec(\mu))$.

Algorithm 2: Frodo.Enc()

Input: Public Key pk, Message μ ;
$\mathbf{S}', \mathbf{E}', \mathbf{E}'' \leftarrow_{\$}$ `generateSmallMatrix()`;
$\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$;
$\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$;
Output: Ciphertext $(\mathbf{C}_1, \mathbf{C}_2) \leftarrow (\mathbf{B}', \mathbf{V} + ec(\mu))$;

Alice can decode this message by calculating $dc(\mathbf{C}_2 - \mathbf{C}_1\mathbf{S})$.

Algorithm 3: Frodo.Dec()

Input: Ciphertext$(\mathbf{C}_1, \mathbf{C}_2)$, Secret Key sk ;
Output: Plaintext $\mu' \leftarrow dc(\mathbf{C}_2 - \mathbf{C}_1\mathbf{S})$;

This gives correct decoding with high probability as

$$\begin{aligned} dc(\mathbf{C}_2 - \mathbf{C}_1\mathbf{S}) &= dc(\mathbf{V} + ec(\mu) - \mathbf{S}'\mathbf{AS} - \mathbf{E}'\mathbf{S}) \\ &= dc(\mathbf{S}'\mathbf{B} + \mathbf{E}'' + ec(\mu) - \mathbf{S}'\mathbf{AS} - \mathbf{E}'\mathbf{S}) \\ &= dc(\mathbf{S}'\mathbf{AS} + \mathbf{S}'\mathbf{E} + \mathbf{E}'' + ec(\mu) - \mathbf{S}'\mathbf{AS} - \mathbf{E}'\mathbf{S}) \\ &= dc(\mathbf{S}'\mathbf{E} + \mathbf{E}'' + ec(\mu) - \mathbf{E}'\mathbf{S}) \end{aligned}$$

So for small enough $\mathbf{E}, \mathbf{E}', \mathbf{E}'', \mathbf{S}, \mathbf{S}'$, we have correct decoding.
For the full detailed description of this algorithm see [6]. A reader willing to become more familiar with encoding and decoding can use the Sage implementation as stated in Appendix A.1.

Now I explain how this relates to the LWE problem. Stated extremely simply, the LWE problem is solving an m-dimensional system of linear equations modulo q in n variables, where a small random portion of noise is added to each equation. To see that the security of FrodoKEM relies on the (later to be formalized) LWE problem, a reduction of FrodoKEM to the LWE-problem is stated.

Proposition 1. An algorithm that solves the LWE-problem, breaks Frodo Encryption.

Proof. An algorithm can break the system, if given the public key $pk = (\mathbf{A}, \mathbf{B})$ it can output the secret key $sk = \mathbf{S}$. Let $((\mathbf{A}, \mathbf{B}), \mathbf{S}) \leftarrow \text{Frodo.KeyGen}(\mathbf{A})$. Split the matrices $\mathbf{B}, \mathbf{S}, \mathbf{E}$ into $\bar{n}$ column vectors with n entries:

$$\begin{pmatrix} \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_{\bar{n}} \end{pmatrix} = \mathbf{A} \begin{pmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \dots & \mathbf{s}_{\bar{n}} \end{pmatrix} + \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \dots & \mathbf{e}_{\bar{n}} \end{pmatrix}$$

Now for each i, $\mathbf{b}_i = \mathbf{A}\mathbf{s}_i + \mathbf{e}_i$ is a system of n equations in n variables with some small error $e_{i,j}$ in each equation, which in fact is the LWE problem. Hence, an adversary capable of solving the LWE problem is capable of finding the secret key sk of the system, which concludes the reduction. □

Attacking Frodo using exhaustive search. An interested reader may wonder how hard it is to simply search through all possible secrets. The following paragraph explains that this is very hard for parameters used in practice. However, it might be somewhat confusing at this point in the thesis, as some results from later in the thesis are already used to describe this. For the proposed parameters for 128-bit security in [6] ($n = 640, q = 2^{15}, \sigma = 2.8$), we see that exhaustive search with success probability ϵ as described in Theorem 44 needs

$$\begin{aligned} m &= \frac{\log(1-\epsilon) - n\log(2t\alpha q + 1)}{\log(2t\alpha)} \\ &= \frac{\log(1-\epsilon) - 640\log(2 \cdot 3\sqrt{\log 640} \cdot 2.8 \cdot \sqrt{2\pi} + 1)}{\log(2 \cdot 3\sqrt{\log 640} \cdot \frac{2.8 \cdot \sqrt{2\pi}}{2^{15}})} \\ &= \frac{\log(1-\epsilon)}{\log(2 \cdot 3\sqrt{\log 640} \cdot \frac{2.8 \cdot \sqrt{2\pi}}{2^{15}})} - \frac{640\log(2 \cdot 3\sqrt{\log 640} \cdot 2.8 \cdot \sqrt{2\pi} + 1)}{\log(2 \cdot 3\sqrt{\log 640} \cdot \frac{2.8 \cdot \sqrt{2\pi}}{2^{15}})} \\ &\approx \frac{\log(1-\epsilon)}{-7.99} + 562 \end{aligned} \tag{1}$$

samples. For $\epsilon = 0.99$, the first term rounded up is 1, thus one needs $563 + 640$ samples and the attack has time complexity $1203(2 \cdot 3\sqrt{\log 640} \cdot 2.8 \cdot \sqrt{2\pi} + 1)^{640} \cdot 1280 \approx 2^{4509}$. This is really far from the best known attack that is mentioned in the paper of Frodo, which has time complexity 2^{145}. An alternative approach could be taking smaller values for ϵ and repeating this experiment until a solution is found. However, the first term in equation 1 gets smaller by decreasing ϵ and this term will always fall between 0 and 1. Hence, this has a negligible effect on the number of samples needed and hence a negligible effect on the runtime. On the other hand, Increasing ϵ to approach 1 does increase the expected runtime.

3.2 Introduction to the LWE problem.

Many presumably quantum-proof cryptographic systems are based on the Learning with Errors problem [46]. The original paper on the LWE problem was published by Oded Regev in 2009 [45].

In the LWE problem, an adversary is challenged to solve a m-dimensional system of linear equations in n variables, with some modulus. The constant term in each equation is called the error term. In this problem the error distribution χ of the error term plays a major role. Every probability distribution that is centered around zero can be used for this. Typically, one uses for χ the gaussian distribution with mean $\mu = 0$ and standard deviation σ, where each sample is rounded to the nearest integer.

Definition 2 *(LWE-Distribution).* Let $n, q \in \mathbb{Z}$ and χ a probability distribution. Given a secret $\mathbf{s} \in \mathbb{Z}_q^n$, denote by $L_{n,q,\chi}$ the LWE-Distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$. That is, the probability distribution by taking $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, e according to χ and output sample $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.

Remark. In other literature, it is sometimes common to define the gaussian distribution based on a parameter α such that the standard deviation of the gaussian distribution is $\alpha q/\sqrt{2\pi}$.

Definition 3 *(Search LWE-problem).* Given m LWE-samples $(\mathbf{a_i}, \mathbf{a_i} \cdot \mathbf{s} + e_i)$, we form a matrix and vector where each sample corresponds to a row, i.e.:

$$(\mathbf{A}, \mathbf{As} + \mathbf{e}) = \left(\begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \vdots \\ \mathbf{a_m} \end{bmatrix}, \begin{bmatrix} \mathbf{a_1} \cdot \mathbf{s} + e_1 \\ \mathbf{a_2} \cdot \mathbf{s} + e_2 \\ \vdots \\ \mathbf{a_m} \cdot \mathbf{s} + e_m \end{bmatrix} \right) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m.$$

An algorithm that given this pair can output the vector $\mathbf{s} \in \mathbb{Z}_q^n$, is said to solve the Search LWE-problem.

Example 4. An example of the LWE problem for $q = 101, n = 5, m = 7$ and χ the rounded gaussian distribution with $\sigma = 1$ is solving the following:

$$\begin{pmatrix} 2 & 86 & 96 & 39 & 57 \\ 15 & 73 & 62 & 6 & 96 \\ 56 & 22 & 4 & 26 & 19 \\ 10 & 68 & 92 & 83 & 51 \\ 5 & 63 & 56 & 47 & 38 \\ 80 & 59 & 99 & 63 & 55 \\ 28 & 88 & 46 & 96 & 74 \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{pmatrix} \approx \begin{pmatrix} 65 \\ 49 \\ 24 \\ 99 \\ 59 \\ 26 \\ 8 \end{pmatrix}$$

It has solution $\mathbf{s} = (60, 63, 11, 8, 4)^T$. Once this solution is found, one can deduce with simple linear algebra that the error term was $\mathbf{e} = (1, 1, 0, 1, -2, -2, 1)^T$.

An implementation of this problem in Sage can be found in appendix A.2. The hardness of this problem lies in the small error term. If not for the error term $\mathbf{e}$, one could simply apply gaussian elimination to solve this problem. However, the small error term yields huge differences in the solution.

Example 5. Suppose I try to solve the above problem, ignoring the small error. In order to apply gaussian elimination, I only need n samples, so I discard the latter $m - n$. A quick Sage calculation yields the following:

```
sage: A
[ 2 86 96 39 57 65]
[15 73 62  6 96 49]
[56 22  4 26 19 24]
[10 68 92 83 51 99]
[ 5 63 56 47 38 59]
```

```
sage: A.echelon_form()
[ 1  0  0  0  0 15]
[ 0  1  0  0  0 97]
[ 0  0  1  0  0 73]
[ 0  0  0  1  0 19]
[ 0  0  0  0  1 16]
```

From this I conclude $\mathbf{s} = (15, 97, 73, 19, 16)^T$. Comparing this with the actual secret from which I generated the samples, I see that only a small error term resulted in a huge mistake in the solution.

3.3 Variations on the LWE problem.

The LWE problem occurs in several slightly different forms in the literature. The following section is used to define some forms of the problem, and to sketch the resemblance between the different versions.

3.3.1 Search LWE versus Decision LWE.

A slightly different version of the search LWE problem is the decision LWE problem.

Definition 6 *(Decision LWE-problem).* Given m pairs $(\mathbf{a_i}, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ either all from the LWE-distribution $L_{n,q,\chi}$ or from the uniformly random distribution: $\mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)$. An algorithm is said to solve the decision LWE-problem if it can distinguish whether the samples were uniformly random samples or LWE-samples.

Definition 7 *((Adversary's) Advantage).* The advantage ϵ of an adversary $\mathcal{A}$ in a distinguishing attack between outcome $L_{n,q,\chi}$ and $\mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)$ is defined

$$\epsilon = \left|\Pr\left[\mathcal{A}(L^m_{n,q,\chi}) | L^m_{n,q,\chi}\right] - \Pr\left[\mathcal{A}(L^m_{n,q,\chi}) | \mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)^m\right]\right|$$

where $\Pr\left[\mathcal{A}(L^m_{n,q,\chi}) | \mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)^m\right]$ should be interpreted as the probability that the adversary decides that the samples are from the LWE distribution, given samples from the uniform distribution.

Remark. An advantage of 0 implies that an adversary can do no better than random guessing, an advantage of 1 implies that an adversary can perfectly distinguish between the cases.

Given a pair $(\mathbf{a}, b)$ from the LWE-distribution and $x, y \in \mathbb{Z}_q$ uniformly random. Denote by s_i the i-th coordinate of $\mathbf{s}$. Consider the pair

$$(\mathbf{a}', b') := \big(\quad (a_1, a_2, \ldots, a_{i-1}, a_i + x, a_{i+1}, \ldots, a_n) \quad , \quad b + xy \quad \big).$$

Consider two cases:

- $y = s_i$: Then the pair $(\mathbf{a}', b')$ is a pair from the LWE-distribution.
- $y \neq s_i$: If q is a prime, then the pair $(\mathbf{a}', b')$ is a uniformly random sample, as $\mathbf{a}', b$ are uniformly random, and multiplication by y maps uniformly random samples to uniformly random samples.

Proposition 8. Let $2 \leq q \leq \text{poly}(n)$ be a prime number. An algorithm that solves the decision LWE problem in polynomial time, can solve the search LWE problem in polynomial time.

Proof. Given m LWE samples $(\mathbf{a_i}, b_i)$. For each of the n coordinates of $\mathbf{s}$ and each pair $(\mathbf{a_i}, b_i)$, feed the m transformations $(\mathbf{a'_i}, b'_i) = (\mathbf{a_i} + (0, \ldots, 0, x, 0, \ldots, 0), b_i + xy)$ as above to the algorithm that solves the decision LWE problem for all q possible values of y. For each coordinate j, let y'_j be the value of y for which the algorithm decides that these are LWE-samples. Pick $s_j = y'_j$ for each j, and the secret $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ is recovered in polynomial time. □

For this proof, q is required to be prime to ensure that $b + xy$ is uniformly random for $y \neq s_i$. In practice, a composite q will also suffice for this property: see lemma 1 of [9] or lemma 3.3 of [42].

Proposition 9. Given an adversary that solves search LWE, an adversary can solve decision LWE.

Proof. Suppose an adversary is given access to an algorithm that given m LWE samples returns the secret vector $\mathbf{s}$. Given m samples $(\mathbf{a_i}, b_i)$, either LWE or uniformly random, and feed these to our algorithm. If it returns a secret $\mathbf{s}$, do the following m calculations: $e_i = \mathbf{a_i} \cdot \mathbf{s} - b_i$ and checks whether this was sampled from the error distribution or not. If it was sampled from the error distribution, the samples were LWE samples, if the algorithm returns no vector $\mathbf{s}$ or the values e_i are not from the error distribution, the adversary was given uniformly random samples. □

3.3.2 Secret Distribution.

In the definition of this problem, the secret $\mathbf{s}$ was chosen from the uniform distribution. At first sight, taking the secret $\mathbf{s}$ from the uniform distribution on $\mathbb{Z}_q^n$ yields a safer system than taking the secret $\mathbf{s}$ sampled from χ. This is because there are fewer possibilities for each entry in the latter distribution and hence fewer possible secrets. It turns out that this is not the case, see chapter 3 of [46]. Intuitively, this can be viewed in the following way: given n samples of the LWE-distribution in matrix form, $(\mathbf{A}, \mathbf{b})$, we have $\mathbf{s} = \mathbf{A}^{-1}(\mathbf{b} - \mathbf{e})$. Hence finding the error (which is sampled from χ) is equivalent to finding the secret. Note that if $\mathbf{A}$ is not invertible, one could replace samples until it is.

If there is a reference to LWE or uniform secret LWE, the LWE problem has a secret distributed under a uniform distribution. If the secret is drawn from the same distribution as the error, it is called (Hermite) normal form LWE. A third form of LWE that can be found in literature, is small-secret LWE. In this version each coordinate of the secret is drawn from a (small) set, say $\{-1, 0, 1\}$. We could either have that the probability of sampling a 1 or a -1 equals p, the probability of sampling a 0 is $1 - 2p$ for some $p \in [0, \frac{1}{2}]$, or that the vector $\mathbf{s}$ has a fixed Hamming weight h (i.e. the number of nonzero entries of $\mathbf{s}$).

Reduction from Uniform Secret LWE to LWE normal form. For simplicity, suppose matrix multiplication can be done in $\mathcal{O}(n^3)$ operations (actually, one can do better, see for example [52]). In lemma 1 of [5] one could find a way to reduce a uniform secret LWE problem to a problem in LWE normal form:

Lemma 10. At the cost of $2n^2$ operations, loss of $\approx n$ uniformly random samples and a precomputation of $\mathcal{O}(n^3)$, one could construct n LWE normal form samples.

Proof. Given n uniform secret LWE samples,

$$(\mathbf{A}, \mathbf{b}) = (\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e}) \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^n$$

If the matrix $\mathbf{A}$ is not invertible, replace samples until it is. Then precompute $\mathbf{A^{-1}}$, at the cost of $\mathcal{O}(n^3)$. Now take n new uniformly random samples:

$$(\mathbf{A}', \mathbf{b}') = (\mathbf{A}', \mathbf{A}' \cdot \mathbf{s} + \mathbf{e}') \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^n$$

Compute

$$\begin{aligned} \mathbf{A}'\mathbf{A}^{-1}\mathbf{b} - \mathbf{b}' &= \mathbf{A}'\mathbf{A}^{-1}(\mathbf{A}\mathbf{s} + \mathbf{e}) - \mathbf{A}'\mathbf{s} - \mathbf{e}' \\ &= \mathbf{A}'\mathbf{s} + \mathbf{A}'\mathbf{A}^{-1}\mathbf{e} - \mathbf{A}'\mathbf{s} - \mathbf{e}' \\ &= \mathbf{A}'\mathbf{A}^{-1}\mathbf{e} - \mathbf{e}' \end{aligned}$$

Then n small secret samples are constructed:

$$(\mathbf{A}'\mathbf{A}^{-1}, \mathbf{A}'\mathbf{A}^{-1}\mathbf{b} - \mathbf{b}') = (\mathbf{A}'\mathbf{A}^{-1}, \mathbf{A}'\mathbf{A}^{-1}\mathbf{e} - \mathbf{e}')$$

These are normal form LWE samples, as $\mathbf{e}$ is the error vector from the LWE samples. Note that the error distribution is symmetric around zero, hence the minus sign doesn't affect the distribution of the error. □

Remark. Solving the new normal form LWE problem yields the solution of the original problem, as the secret of this new problem is exactly the error term of the original uniform secret LWE problem. With this error term we can calculate $\mathbf{s} = \mathbf{A}^{-1}(\mathbf{b} - \mathbf{e})$ and recover the secret of the original problem.

More on secret distribution and equivalence between the several options can also be found for example on page 600 of [9].

3.4 Ring-LWE.

For a cryptographic system, at least n LWE samples $(\mathbf{a}, b) = ((a_0, a_1, \ldots, a_{n-1}), b) \in \mathbb{Z}_q^n \times \mathbb{Z}$ are required. This implies that public key sizes are of order at least n^2. A possibility to reduce this, is to add some structure in the public key. Define an operation

$$\sigma : \quad (a_0, \ldots, a_{n-1}) \quad \mapsto \quad (-a_{n-1}, a_0, \ldots, a_{n-2}).$$

One could take only one LWE sample $(\mathbf{a}, b) = ((a_0, a_1, \ldots, a_{n-1}), b)$, and take for the $n-1$ samples the following: $(\mathbf{a_i}, b_i) = (\sigma^i(\mathbf{a}), \sigma^i(\mathbf{a}) \cdot \mathbf{s} + e_i)$ with e_i distributed according to the error distribution. In the following paragraph I explain that this is equivalent to replacing the ring $\mathbb{Z}_q^n$ with $\mathbb{Z}_q/\langle x^n + 1\rangle$.

Let $f(x) = x^n + 1$, where n is a pure power of 2, to ensure this polynomial is cyclotomic. A cyclotomic polynomial has several nice properties that have been proven useful for cryptographic purposes [38]. Some more on the choice of polynomial is found in Section 6.1. With this particular form, the polynomial is maximally sparse and polynomial arithmetic modulo $f(x)$ can be done efficiently. We let $q \equiv 1 \mod 2n$ a sufficiently large prime modulus, to ensure that the noise is small relative to the modulus.

Let $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$ denote the ring of polynomials with coefficients in $\mathbb{Z}_q$ modulo $f(x)$. Analogous to the LWE problem, let $s = s(x) \in R_q$ be the secret, and let a RLWE sample consist of a pair $(a, a \cdot s + e)$, where $a \in R_q$ uniformly random, and each coordinate of e independent identically distributed by χ. The original article on RLWE was written in 2010 by Lyubashevsky, Peikert and Regev [37].

Example 11. Take $n = 4$, $q = 17$. Then an example of the RLWE problem is solving the following equation in $R_{17} = \mathbb{Z}_{17}/\langle x^4 + 1\rangle$

$$(6x^3 + 6x^2 + 4x + 11) \cdot (s_3x^3 + s_2x^2 + s_1x + s_0) \approx (9x^3 + 2x^2 + 4x + 1)$$

Here, the solution would be $s = 15x^3 + 12x^2 + 13x + 10$, and the small error term would be $e = -2x^3 + x^2 - 1$.

To sketch the reduction from RLWE to LWE with the samples as earlier in this paragraph, I wrote out the two for a small n, e.g. 4: Then LWE is instantiated by $(\mathbf{A}, \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$:

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ -a_3 & a_0 & a_1 & a_2 \\ -a_2 & -a_3 & a_0 & a_1 \\ -a_1 & -a_2 & -a_3 & a_0 \end{pmatrix} \cdot \begin{pmatrix} s_3 \\ s_2 \\ s_1 \\ s_0 \end{pmatrix} \approx \begin{pmatrix} b_3 \\ b_2 \\ b_1 \\ b_0 \end{pmatrix}$$

From which I obtain the following equations:

$$\begin{aligned} a_0s_3 + a_1s_2 + a_2s_1 + a_3s_0 &\approx b_3 \\ -a_3s_3 + a_0s_2 + a_1s_1 + a_2s_0 &\approx b_2 \\ -a_2s_3 - a_3s_2 + a_0s_1 + a_1s_0 &\approx b_1 \\ -a_1s_3 - a_2s_2 - a_3s_1 + a_0s_0 &\approx b_0 \end{aligned} \tag{2}$$

Conversely, RLWE is instantiated by polynomials $(a(x), a(x)s(x) + e(x))$, where $a(x) = a_0 +$

$a_1x + a_2x^2 + a_3x^3 \in R_q$, $s(x) = s_0 + s_1x + s_2x^2 + s_3x^3 \in R_q$. Then

$$\begin{aligned}
a(x)s(x) =&(a_0 + a_1x + a_2x^2 + a_3x^3)(s_0 + s_1x + s_2x^2 + s_3x^3) \\
=&a_0s_0 + a_0s_1x + a_0s_2x^2 + a_0s_3x^3 + a_1s_0x + a_1s_1x^2 + a_1s_2x^3 + a_1s_3x^4 \\
&+ a_2s_0x^2 + a_2s_1x^3 + a_2s_2x^4 + a_2s_3x^5 + a_3s_0x^3 + a_3s_1x^4 + a_3s_2x^5 + a_3s_3x^6 \\
=&a_0s_0 + a_0s_1x + a_0s_2x^2 + a_0s_3x^3 + a_1s_0x + a_1s_1x^2 + a_1s_2x^3 - a_1s_3 \\
&+ a_2s_0x^2 + a_2s_1x^3 - a_2s_2 - a_2s_3x + a_3s_0x^3 - a_3s_1 - a_3s_2x - a_3s_3x^2 \\
=&(a_0s_0 - a_1s_3 - a_2s_2 - a_3s_1) + (a_0s_1 + a_1s_0 - a_2s_3 - a_3s_2)x \\
&+ (a_0s_2 + a_1s_1 + a_2s_0 - a_3s_3)x^2 + (a_0s_3 + a_1s_2 + a_1s_2 + a_2s_1 + a_2s_0)x^3 \\
\approx&b_0 + b_1x + b_2x^2 + b_3x^3
\end{aligned} \tag{3}$$

If I compare the coefficients of equation 3, I perfectly obtain the equations in 2. A simple implementation of RLWE can be found in appendix A.3.

4 Lattices.

The most efficient known attacks to the LWE problem make use of the fact that solving the LWE problem can be interpreted as certain lattice problems. I use this section to explain how to reduce the LWE problem to several lattice problems. I mainly follow the approach of [15].

4.1 Lattices.

In order to discuss lattices, a few definitions are stated.

Definition 12. An m-dimensional lattice $\Lambda \subset \mathbb{Z}^m$ is a full-rank additive subgroup of $\mathbb{Z}^m$.[1] The distance between a lattice Λ and a point $\mathbf{a} \in \mathbb{R}^m$ is defined $d(\mathbf{a}, \Lambda) := \min_{\mathbf{x}\in\Lambda} ||\mathbf{a} - \mathbf{x}||$. Furthermore $\lambda_1(\Lambda) := \min_{\mathbf{a}\in\Lambda\setminus\{\mathbf{0}\}} ||\mathbf{a}||$ is the length of a smallest nonzero vector in the lattice.

Definition 13 *(Bounded Distance Decoding Problem).* Given a lattice Λ, a point $\mathbf{a} \in \mathbb{Z}^m$ such that $d(\mathbf{a}, \Lambda) < \frac{1}{2}\lambda_1(\Lambda)$. The Bounded Distance Decoding (BDD) problem is finding the lattice point $\mathbf{x} \in \Lambda$ closest to $\mathbf{a}$.

In practice, the hardness of the bounded distance decoding problem increases rapidly as the dimension grows. For illustrational purposes, an example of the BDD problem in two dimensions is given which is considered computationally "easy".

Example 14. Given the lattice $\Lambda := \left\{ a \cdot \begin{pmatrix} 2 \\ 2 \end{pmatrix} + b \cdot \begin{pmatrix} 2 \\ -2 \end{pmatrix} | a, b \in \mathbb{Z} \right\} \subset \mathbb{R}^2$, see figure 1. The point $\mathbf{a}$ is closer than half the minimum distance to the lattice, and hence the Bounded Distance Decoding problem is reducing this point to the point $(2, 2) \in \Lambda$, the closest point to $\mathbf{a}$ in the lattice.

LWE problem in lattices. Definitions 15 and 16 form the basis of the connection between Lattices and the LWE problem.

Definition 15. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$ with $m \geq n$ such that the first n rows form an invertible matrix, we define the lattice

$$\mathcal{L}(\mathbf{A}) := \{ \quad \mathbf{z} \in \mathbb{Z}^m \quad | \quad \exists \mathbf{s} \in \mathbb{Z}^n : \quad \mathbf{z} = \mathbf{A}\mathbf{s} \mod q \quad \} \quad \subset \mathbb{Z}^m$$

Note that indeed this produces a subgroup: as $\mathbf{z}, \mathbf{z}' \in \mathcal{L}(\mathbf{A})$ implies $\mathbf{z} = \mathbf{A}\mathbf{s} \mod q, \mathbf{z}' = \mathbf{A}\mathbf{s}' \mod q$ for some $\mathbf{s}, \mathbf{s}' \in \mathbb{Z}^m$. Therefore $\mathbf{z} + \mathbf{z}' = \mathbf{A}\mathbf{s} + \mathbf{A}\mathbf{s}' = \mathbf{A}(\mathbf{s} + \mathbf{s}') \mod q$, and $\mathbf{s} + \mathbf{s}' \in \mathbb{Z}^n$. Hence $\mathcal{L}(\mathbf{A})$ is closed under addition. Trivially associativity holds, $\mathbf{0} \in \mathcal{L}(\mathbf{A})$ and for any $\mathbf{z} \in \mathcal{L}(\mathbf{A})$ we also have $-\mathbf{z} \in \mathcal{L}(\mathbf{A})$.
To show that this defines indeed a lattice, it remains to check that it is full-rank (i.e. m) . To do so, split the matrices into blocks. Let $\mathbf{A} = \begin{pmatrix} \mathbf{A_1} \\ \mathbf{A_2} \end{pmatrix}$ with $\mathbf{A_1} \in \mathbb{Z}_q^{n\times n}, \mathbf{A_2} \in \mathbb{Z}_q^{(m-n)\times n}$ where we suppose $\mathbf{A_1}$ is invertible. Similar to this, $\mathbf{z} = \begin{pmatrix} \mathbf{z_1} \\ \mathbf{z_2} \end{pmatrix}$ with $\mathbf{z_1} \in \mathbb{Z}_q^n, \mathbf{z_2} \in \mathbb{Z}_q^{m-n}$. Suppose a point lies in our lattice: $\mathbf{z} \in \mathcal{L}(\mathbf{A})$. Then, by definition there exists $\mathbf{s} \in \mathbb{Z}_q^n$ such that $\mathbf{z} = \mathbf{A}\mathbf{s} \mod q$. In block form, such a point satisfies both

$$\begin{aligned} \mathbf{z_1} &= \mathbf{A_1}\mathbf{s} \mod q \\ \mathbf{z_2} &= \mathbf{A_2}\mathbf{s} \mod q. \end{aligned}$$

But then $\mathbf{s} = {\mathbf{A_1}}^{-1}\mathbf{z_1} \mod q$ and hence $\mathbf{z_2} = \mathbf{A_2}{\mathbf{A_1}}^{-1}\mathbf{z_1} \mod q$. From the latter equation, I conclude that there exists a $\mathbf{v} \in \mathbb{Z}_q^{m-n}$ such that

$$\mathbf{z_2} = \mathbf{A_2}{\mathbf{A_1}}^{-1}\mathbf{z_1} + q\mathbf{v}$$

[1] In other literature, it is common to define a lattice as a subgroup of $\mathbb{R}^m$. For the purposes of this thesis, it suffices to define a lattice as a subgroup of $\mathbb{Z}^m$.

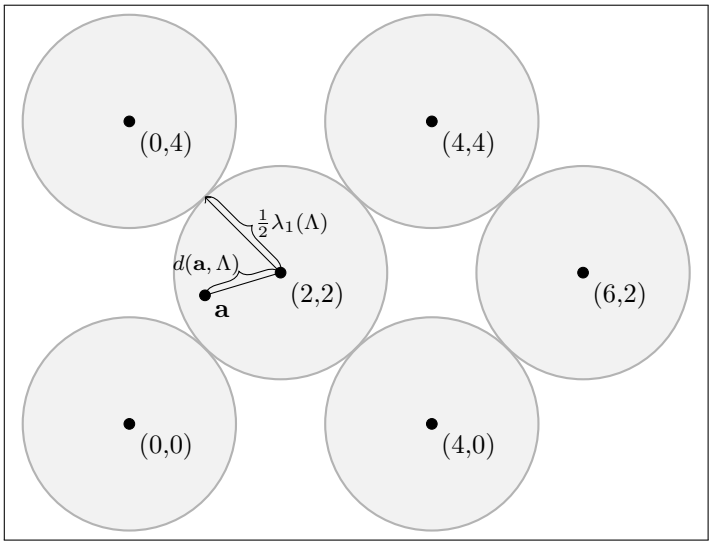


Figure 1: The Bounded Distance Decoding problem visually.

Now the columns of the following matrix form a basis for $\mathcal{L}(\mathbf{A})$:

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathbf{A_2}{\mathbf{A_1}}^{-1} & q\mathbf{I}_{m-n} \end{pmatrix} \tag{4}$$

As

$$\mathbf{B}\begin{pmatrix} \mathbf{z_1} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{z_1} \\ \mathbf{A_2}{\mathbf{A_1}}^{-1}\mathbf{z_1} + q\mathbf{v} \end{pmatrix} = \begin{pmatrix} \mathbf{z_1} \\ \mathbf{z_2} \end{pmatrix}$$

I conclude that $\mathbf{B}$ is indeed an m-dimensional basis of $\mathcal{L}(\mathbf{A})$.

Suppose one is given m LWE samples in matrix form: $(\mathbf{A}, \mathbf{As} + \mathbf{e}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$. Then $\mathbf{x} = \mathbf{As}$ is an element of the lattice $\mathcal{L}(\mathbf{A})$. As the error term follows a gaussian distribution, the error is very likely to fall into the first several standard deviations away from the mean. For LWE parameters typically used for cryptographic purposes, this is significantly smaller than $\lambda_1(\mathcal{L}(\mathbf{A}))$ [15]. Therefore, with high probability the closest lattice point to $\mathbf{As}+\mathbf{e}$ is $\mathbf{x}$. This can be seen as a BDD problem. If one is able to solve this, the error term in the original problem is eliminated and hence if $\mathbf{A}$ is invertible one is able to recover the secret vector $\mathbf{s}$ and thus solve the search LWE problem.

Definition 16. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, define the lattice

$$\mathcal{L}^\perp(\mathbf{A}) := \{ \quad (\mathbf{y}^T||\mathbf{z}^T)^T \in \mathbb{Z}^n \times \mathbb{Z}^m \quad | \quad \mathbf{y}^T = \mathbf{z}^T\mathbf{A} \mod q \quad \} \quad \subset \mathbb{R}^{m+n}$$

Proposition 17. The dual lattice $\mathcal{L}^\perp(\mathbf{A})$ is generated by the columns of

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_n & \mathbf{A}^T \\ \mathbf{0}_{m\times n} & \mathbf{I}_m \end{pmatrix}.$$

Proof. Let

$$\Lambda(\mathbf{B}) := \left\{ \mathbf{B}\begin{pmatrix} \mathbf{b_1} \\ \mathbf{b_2} \end{pmatrix} \quad | \quad \mathbf{b_1} \in \mathbb{Z}^n, \mathbf{b_2} \in \mathbb{Z}^m \right\} = \left\{ \begin{pmatrix} q\mathbf{b_1} + \mathbf{A}^T\mathbf{b_2} \\ \mathbf{b_2} \end{pmatrix} \quad | \quad \mathbf{b_1} \in \mathbb{Z}^n, \mathbf{b_2} \in \mathbb{Z}^m \right\}$$

To show that any $\begin{pmatrix} \mathbf{v} \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} q\mathbf{b_1} + \mathbf{A}^T\mathbf{b_2} \\ \mathbf{b_2} \end{pmatrix} \in \Lambda(\mathbf{B})$ is also an element of $\mathcal{L}^\perp(\mathbf{A})$, consider

$$\begin{aligned} \mathbf{v}^T - \mathbf{w}^T\mathbf{A} &= (q\mathbf{b_1} + \mathbf{A}^T\mathbf{b_2})^T - \mathbf{b_2}^T\mathbf{A} \\ &= q\mathbf{b_1}^T + \mathbf{b_2}^T\mathbf{A} - \mathbf{b_2}^T\mathbf{A} \\ &= q\mathbf{b_1}^T \end{aligned}$$

Hence I conclude that $\mathbf{v}^T = \mathbf{w}^T\mathbf{A} \mod q$, and thus $\Lambda(\mathbf{B}) \subseteq \mathcal{L}^\perp(\mathbf{A})$. On the other hand let $(\mathbf{v}^T||\mathbf{w}^T)^T \in \mathcal{L}^\perp(\mathbf{A})$, thus there exists $\mathbf{x} \in \mathbb{Z}^n$ such that

$$\mathbf{v}^T = \mathbf{w}^T\mathbf{A} + q\mathbf{x}^T. \tag{5}$$

Pick $\mathbf{b_1} = \mathbf{x}, \mathbf{b_2} = \mathbf{w}$. Then

$$\mathbf{B}\begin{pmatrix} \mathbf{b_1} \\ \mathbf{b_2} \end{pmatrix} = \mathbf{B}\begin{pmatrix} \mathbf{x} \\ \mathbf{w} \end{pmatrix} = \begin{pmatrix} q\mathbf{x} + \mathbf{A}^T\mathbf{w} \\ \mathbf{w} \end{pmatrix}$$

Combining with the transposed version equation 5 I conclude that $(\mathbf{v}^T||\mathbf{w}^T)^T \in \Lambda(\mathbf{B})$. Thus $\mathcal{L}^\perp(\mathbf{A}) \subseteq \Lambda(\mathbf{B})$, and hence conclude that $\mathcal{L}^\perp(\mathbf{A})$ is spanned by the columns of $\mathbf{B}$. □

Definition 18 *(Short Integer Solution Problem).* Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, the short integer solution (SIS) problem is finding a vector $\mathbf{0} \neq \mathbf{v} \in \mathbb{Z}^m$ such that $\mathbf{v}^T\mathbf{A} = \mathbf{0} \mod q$ and $||\mathbf{v}|| \leq \beta$ for a given $\beta < q \in \mathbb{Z}$.

Proposition 19. An algorithm that solves the BDD problem solves the SIS problem.

Idea behind Proof. Given basis $\mathbf{B} := \{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$, define $\mathbf{B}^i := \{\mathbf{b_1}, \ldots, \mathbf{b_{i-1}}, 2\mathbf{b_i}, \ldots, \mathbf{b_n}\}$. For each $1 \leq i \leq n$, solve the BDD problem for lattice $\mathcal{L}^\perp(\mathbf{B}^i)$ and vector $\mathbf{b_i}$ and call the given output vector $\mathbf{x_i}$. Then, pick the smallest vector from the set $\{\mathbf{x_i} - \mathbf{b_i}\}$ to obtain the smallest vector in the lattice $\mathbf{B}$. □

As this proposition is only used to describe the similarity between the BDD and SIS problem and the complete proof doesn't add any relevant information for the purpose of this thesis, the details are omitted. An interested reader can find them in [39].

Proposition 20. Given an algorithm that solves the SIS problem and an algorithm that distinguishes m Gaussian samples from uniformly random samples, one can solve decision LWE.

Proof. Suppose there are m samples given, written in matrix form $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$, where either $\mathbf{b} = \mathbf{As} + \mathbf{e}$ or $\mathbf{b}$ is uniformly random. Now, by assumption, the algorithm finds a small vector $\mathbf{v} \in \mathcal{L}^\perp(\mathbf{A})$. In case $\mathbf{b}$ is uniformly random, $\mathbf{v} \cdot \mathbf{b}$ is also uniformly random (For composite q, this is not exactly uniformly random, but in practice we can assume it is). In case $\mathbf{b} = \mathbf{As} + \mathbf{e}$, and $\mathbf{v}$ is a lattice point, $\mathbf{v} \cdot \mathbf{b} = \mathbf{v} \cdot \mathbf{As} + \mathbf{v} \cdot \mathbf{e} = \mathbf{v} \cdot \mathbf{e}$. As $\mathbf{e}$ follows a Gaussian distribution, by assumption one could distinguish this from uniform and solve the decision problem. □

4.2 Bounded Distance Decoding.

In the next section, the Gram-Schmidt orthogonalization process is used several times.

Theorem 21 *(Gram-Schmidt Orthogonalization method).* Given a basis $\{\mathbf{b_1}, \ldots, \mathbf{b_m}\}$ of an m-dimensional subspace of $\mathbb{R}^n$, define

$$\begin{aligned}
\mathbf{b_1^*} &= \mathbf{b_1} \\
\mathbf{b_2^*} &= \mathbf{b_2} - \mu_{2,1}\mathbf{b_1} \qquad \text{where } \mu_{2,1} = \tfrac{\mathbf{b_2}\cdot\mathbf{b_1^*}}{\mathbf{b_1^*}\cdot\mathbf{b_1^*}} \\
&\vdots \\
\mathbf{b_m^*} &= \mathbf{b_m} - \textstyle\sum_{i<m} \mu_{m,i}\mathbf{b_i} \quad \text{where } \mu_{m,i} = \tfrac{\mathbf{b_m}\cdot\mathbf{b_i^*}}{\mathbf{b_i^*}\cdot\mathbf{b_i^*}}
\end{aligned}$$

Then $\{\mathbf{b_1^*}, \ldots, \mathbf{b_m^*}\}$ is an orthogonal basis of the same subspace.

Proof. See [44]. □

Remark. These bases span the same subspace over $\mathbb{R}^n$, but these bases do not necessarily span the same lattice.

Babai's Nearest Planes. A simple algorithm for solving the BDD problem is the Babai Nearest Plane Algorithm as described in algorithm 4. There are several claims in this section that are not proven. The literature that is followed in this section are [51] and [26]. The first step of the algorithm is to apply the LLL algorithm to the basis. The reader will become familiar with this concept in section 4.3.1

Algorithm 4: Babai's nearest plane Algorithm

Input: Basis vectors $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}, \mathbf{a} \in \mathbb{Z}_q^n$;
Step 1: Apply the LLL algorithm to the basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$;
Step 2: Compute the corresponding Gram-Schmidt basis $\{\mathbf{b_1^*}, \ldots, \mathbf{b_n^*}\}$;
$\mathbf{a_n} \leftarrow \mathbf{a}$ **for** $i = n$ **to** 1 **do**
  $\ell_i \leftarrow \mathbf{a_i} \cdot \mathbf{b_i^*} / ||\mathbf{b_i^*}||^2$;
  $\mathbf{y_i} \leftarrow \lceil \ell_i \rfloor \mathbf{b_i}$;
  $\mathbf{a_{i-1}} \leftarrow \mathbf{a_i} - (\ell_i - \lceil \ell_i \rfloor)\mathbf{b_i^*} - \lceil \ell_i \rfloor \mathbf{b_i}$;
end
return $\mathbf{b} = \mathbf{y_1} + \ldots + \mathbf{y_n}$;

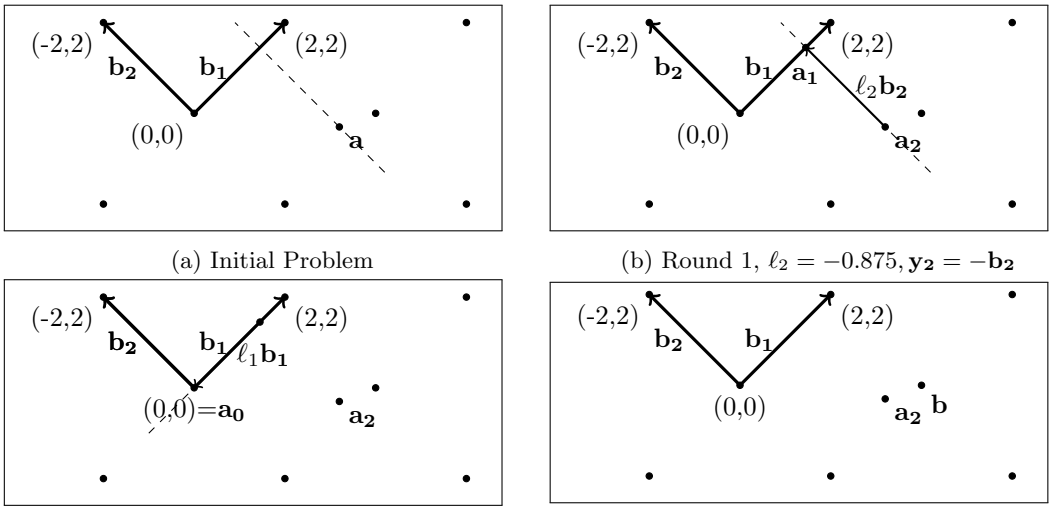
Claim 22 *(Runtime of Babai's algorithm).* Given a basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ of a lattice Λ with $B := \max\{||\mathbf{b_1}||^2, \ldots, ||\mathbf{b_n}||^2\}$ and a point $\mathbf{a} \in \Lambda$. Then algorithm 4 runs in $\mathcal{O}(n^5(\log B)^2)$

Claim 23 *(Correctness of Babai's algorithm).* Given a basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ of Λ and $\mathbf{a} \in \mathbb{R}^n$. If there exist a vector $\mathbf{b} \in \Lambda$ such that $||\mathbf{a} - \mathbf{b}|| \leq \frac{1}{2}\min\{||\mathbf{b_i^*}||\}$, then algorithm 4 outputs $\mathbf{b}$ on input $\mathbf{a}$.

Note that this implies that the decoding of $\mathbf{b} = \mathbf{As} + \mathbf{e}$ to $\mathbf{As}$ is correct if $||\mathbf{As} + \mathbf{e} - \mathbf{As}|| = ||\mathbf{e}|| \leq \frac{1}{2}\min\{||\mathbf{b_i^*}||\}$. The correctness of decoding using this algorithm is thus dependent on the (standard deviation of the) error distribution and the length of the shortest vector in the Gram-Schmidt basis of the lattice.

Proposition 24. Given a basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ of Λ and $\mathbf{a} \in \mathbb{R}^n$. The output $\mathbf{b} \in \Lambda$ of algorithm 4 lies in the parallelepiped

$$\left\{ \mathbf{a} + \sum_{i=1}^{n} \ell_i \mathbf{b_i^*} \quad | \quad \ell_i \in \mathbb{R}, \quad \text{with } |\ell_i| \leq \frac{1}{2} \right\}.$$



(a) Initial Problem

(b) Round 1, $\ell_2 = -0.875, \mathbf{y_2} = -\mathbf{b_2}$

(c) Round 2, $\ell_1 = 0.725, \mathbf{y_1} = \mathbf{b_1}$

(d) Output: $\mathbf{b} = \lceil \ell_1 \rfloor \mathbf{b_1} + \lceil \ell_2 \rfloor \mathbf{b_2}$

Figure 2: Example of Babai's nearest plane algorithm.

Proof. In each step of the algorithm, $\mathbf{a_{i-1}} = \mathbf{a_i} - (\ell_i - \lceil \ell_i \rfloor)\mathbf{b_i^*} - \lceil \ell_i \rfloor \mathbf{b_i}$ (and $\mathbf{a_0} = \mathbf{0}$). Therefore

$$\mathbf{0} = \mathbf{a_0} = \mathbf{a} - \sum_{i=1}^{n} (\ell_i - \lceil \ell_i \rfloor)\mathbf{b_i^*} - \lceil \ell_i \rfloor \mathbf{b_i}$$

As $\mathbf{b} = \sum_{i=1}^{n} \lceil \ell_i \rfloor \mathbf{b_i}$, conclude that

$$\mathbf{b} = \mathbf{a} - \sum_{i=1}^{n} (\ell_i - \lceil \ell_i \rfloor)\mathbf{b_i^*}$$

Where clearly $|\ell_i - \lceil \ell_i \rfloor| \leq \frac{1}{2}$ for each $1 \leq i \leq n$. □

Corollary 25. Given a basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ of Λ and $\mathbf{a} \in \mathbb{R}^n$. The output $\mathbf{b} \in \Lambda$ of algorithm 4 satisfies $||\mathbf{a} - \mathbf{b}|| \leq \frac{1}{4} \sum_{i=1}^{n} ||\mathbf{b_i^*}||$.

Proof. This follows immediately from proposition 24. □

Another way to find a point in a lattice that is close to the target point is the Babai's rounding technique. Let $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ be a basis of lattice $\Lambda \subset \mathbb{Z}^n$, given $\mathbf{a} \in \mathbb{Z}^n$ write $\mathbf{a} = \sum_{i=1}^{n} c_i \mathbf{b_i}$, with $c_i \in \mathbb{R}$ as a linear combination over the real numbers of the basis vector. The babai's rounding technique is rounding each coefficient to the nearest integer: $\mathbf{a'} = \sum_{i=1}^{n} \lceil c_i \rfloor \mathbf{b_i} \in \Lambda$

Claim 26. Given an orthogonal basis of a lattice Λ. Then Babai's nearest plane algorithm and Babai's rounding technique both give a correct solution to the BDD problem.

Example 27. Given lattice $\Lambda := \{a \cdot (2,2)^T + b \cdot (-2,2)^T | a, b \in \mathbb{Z}\}$ and $\mathbf{a} = (3.2, -0.3) \in \mathbb{R}^2$. The steps of the Babai's nearest plane algorithm are given in figure 2.

4.3 Lattice Reduction.

Now that LWE can be considered as a lattice problem, techniques used on lattices can be used to solve the LWE problem. A lattice is not uniquely determined by it's basis: Different bases can refer to the same lattice. Typically, a lattice problem is hard because it doesn't have a nice

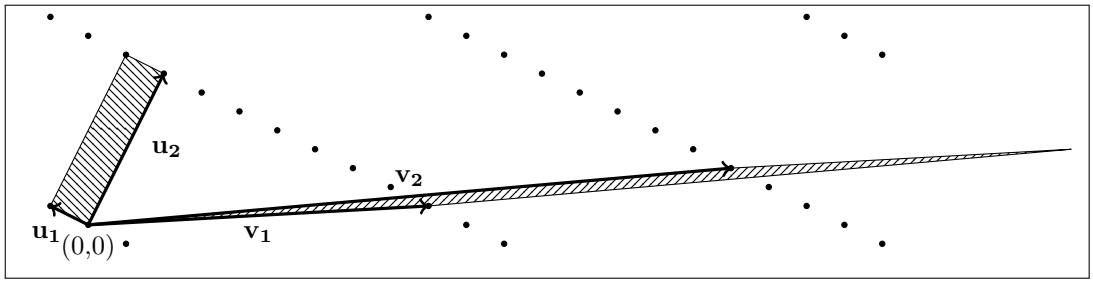


Figure 3: Basis Reduction of a two dimensional lattice and their corresponding fundamental parallelepipeds.

basis. To easily do calculations on lattices , one prefers a lattice with as small as possible basis vectors, as close as perpendicular as possible to each other. Now I formalize what it is to be a good basis.

Definition 28 *(Fundamental paralellepiped).* For a lattice with basis $\mathbf{B} = \{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_n}\}$, the fundamental paralellepiped is defined as

$$P(\mathbf{B}) := \{ \quad \mathbf{B} \cdot \mathbf{x} \quad | \quad \mathbf{x} \in [0,1)^n \quad \}$$

The volume of the fundamental paralellepiped is defined $\text{vol}(P(\mathbf{B})) := \sqrt{\det(\mathbf{B}^T\mathbf{B})}$.

To valuate perpendicularness of a basis, is to compare the product of the length of the basis vectors to the volume of the fundamental paralellepiped they define. This is done by the *Hermite factor*.

Definition 29 *(Hermite factor).* The Hermite factor δ_0^n of a lattice with basis $\mathbf{B}$, which has shortest basis vector $\mathbf{b_1}$, is defined as

$$\delta_0^n = \frac{||\mathbf{b_1}||}{\text{vol}(P(\mathbf{B}))^{1/n}}.$$

δ_0 is referred to as the root Hermite factor, and $\log \delta_0$ as the log root Hermite factor.

Example 30. For a visual representation of this example, see figure 3. Given the lattice spanned (with integer coefficients) by $\mathbf{v_1} = (18,1)^T, \mathbf{v_2} = (34,3)^T$. This basis has Hermite factor $\delta_0^2(\{\mathbf{v_1}, \mathbf{v_2}\}) \approx 4$. A nicer basis for the same lattice would be $\{\mathbf{u_1} = (-2,1)^T, \mathbf{u_2} = (4,8)^T\}$, which has Hermite factor $\delta_0^2(\{\mathbf{u_1}, \mathbf{u_2}\}) = 0.5$ and much smaller basis vectors.

4.3.1 LLL Algorithm.

In 1982, Lenstra, Lenstra and Lovász found an algorithm that reduces a basis of a lattice into a nice basis. They qualify a nice basis if it is an LLL-reduced basis. See [35].

Definition 31 *(LLL-reduced basis).* Let $\mathbf{B} = \{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ a basis for a lattice Λ, $\mathbf{B}^* = \{\mathbf{b_1^*}, \ldots, \mathbf{b_n^*}\}$ its corresponding Gramm-Schmidt orthogonal basis and let $\mu_{i,j} := \frac{\mathbf{b_j} \cdot \mathbf{b_i^*}}{\mathbf{b_i^*} \cdot \mathbf{b_i^*}}$. $\mathbf{B}$ is said to be LLL-reduced if :

1. For all $1 \leq j < i \leq n$ we have $\mu_{i,j} \leq \frac{1}{2}$.
2. For each $1 \leq i < n$, $||\mathbf{b_{i+1}^*} + \mu_{i+1,i}\mathbf{b_i^*}||^2 \geq \frac{3}{4}||\mathbf{b_i^*}||^2$. [2]

LLL-reduced bases have certain nice properties that are stated (not proven) below.

[2] More general definitions of LLL reduced basis can be found in literature, with a parameter $0 < \delta < 1$. For simplicity we use the specific case $\delta = \frac{3}{4}$.

Claim 32. If $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ is a LLL-reduced basis for the lattice Λ, then the following statements hold:

(1) $||\mathbf{b_1}|| \leq 2^{\frac{n-1}{2}} \lambda_1(\Lambda)$.

(2) $\prod_{i=1}^{n} ||\mathbf{b_i}|| \leq 2^{\frac{n(n-1)}{4}} \det(\mathbf{B})$

By this, an LLL-reduced basis has a basis vector that differs at most an exponential factor in the dimension of the lattice from the actual shortest vector in the lattice.

Algorithm 5: LLL Algorithm

Input: Basis vectors $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$;
Step 1: Make the algorithm nearly orthogonal ;
for $i = 1$ **to** n **do**
  for $k = i - 1$ **to** *1* **do**
    $\mathbf{b_i} \leftarrow \mathbf{b_i} - \lfloor \mu_{i,k} \rceil \mathbf{b_k}$
  end
end
Step 2: Check condition (2);
for $i = 1$ **to** $n - 1$ **do**
  if $||\mathbf{b^*_{i+1}} + \mu_{i+1,i}\mathbf{b^*_i}||^2 < \frac{3}{4}||\mathbf{b^*_i}||^2$ **then**
    swap $\mathbf{b_i}$ with $\mathbf{b_{i+1}}$;
    Restart step 1 with this new basis;
  end
end
Output: LLL-reduced basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$;

Claim 33 *(LLL-algorithm).* Given a basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ with $B := \max\{||\mathbf{b_1}||^2, \ldots, ||\mathbf{b_n}||^2\}$. Then algorithm 5 runs in $\mathcal{O}(n^6(\log B)^3)$ and reduces this basis to an LLL-reduced basis. [35]

The LLL algorithm constantly updates the basis of a lattice, while it leaves the lattice it defines unchanged. There are several operations to change a basis, but keep the lattice untouched. Three of them are, given basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$:

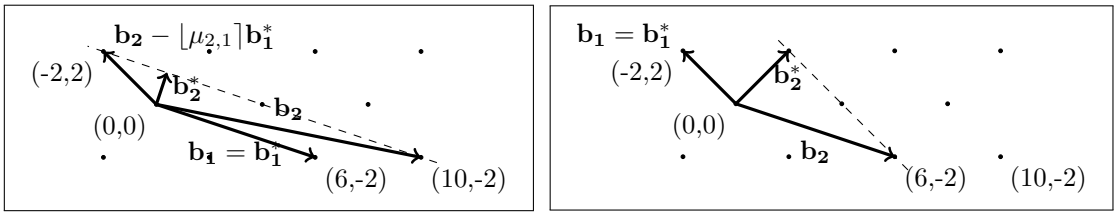
(1) Swap two vectors in the basis.

(2) Replace a basis vector $\mathbf{b_i}$ with $-\mathbf{b_i}$.

(3) Replace a basis vector $\mathbf{b_i}$, with $\mathbf{b_i} + \sum_{j \neq i} a_j \mathbf{b_j}$ where $a_j \in \mathbb{Z}$. That is, add a integer linear combinaton of the other basis vectors to it.

Once again this problem is visually illustrated in two dimensions, but keep in mind that this is not a hard problem in two dimensions.

Example 34. Given lattice Λ spanned by basis $\mathbf{B_0} := \{\mathbf{b_1} := (6, -2)^T, \mathbf{b_2} := (10, -2)^T\} \subset \mathbb{Z}^2$, see figure 4. Compute $\mu_{2,1} = \frac{8}{5} \not\leq \frac{1}{2}$, hence this basis isn't LLL-reduced. Apply the first round of LLL (with corresponding orthogonal basis $\{\mathbf{b_1^*} = (6, -2)^2, \mathbf{b_2^*} = (\frac{2}{5}, \frac{6}{5})\}$, and replace $\mathbf{b_2}$ with $\mathbf{b_2} - 2\mathbf{b_1}$ to obtain the new basis $\{\mathbf{b_1} := (6, -2)^T, \mathbf{b_2} = (-2, 2)^T\}$. This basis however, doesn't satisfy the second condition, and thus both basis vectors are swapped. We start the second round with basis $\mathbf{B_1} = \{(-2, 2)^T, (6, -2)^T\}$, with corresponding orthogonal base $\{(-2, 2)^T, (2, 2)^T\}$. As this vector is also in the lattice, the basis is updated to the exact same basis. It turns out that this basis satisfies the conditions, which may not be a surprise as this basis was orthogonal.



(a) First round of the LLL-algorithm

(b) Second round of the LLL-algorithm

Figure 4: LLL algorithm in two dimensions

```
1 sage: load('primalattack.sage')
2 sage: s = generateSecret()
3 sage: A = generateSamples()
4 sage: E = generateError(1)
5 sage: b = A*s + E
6 sage: recoverSecret(A,b)
7 [72]
8 [46]
9 [70]
```

Listing 1: Solving a small scale LWE-problem

Example 35. the following Sage code is an example of solving the LWE problem with Babai's nearest plane method and LLL-basis reduction. For the file `primalattack.sage`, see the code in appendix A.4. However, without seeing the code, one can intuitively guess what each Sage call in Listing 1 should do. We start with our public key:

$$pk = (\mathbf{A}, \mathbf{As} + \mathbf{e}) = \left(\begin{bmatrix} 95 & 73 & 82 \\ 98 & 21 & 94 \\ 26 & 36 & 81 \\ 55 & 94 & 68 \\ 13 & 79 & 11 \end{bmatrix}, \begin{bmatrix} 82 \\ 60 \\ 7 \\ 16 \\ 87 \end{bmatrix} \right)$$

From which one makes the lattice spanned by the rows of matrix $\mathbf{B}$ (and/or corresponding LLL-reduced basis $\mathbf{B}'$)

$$\mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 30 & 39 \\ 0 & 1 & 0 & 94 & 24 \\ 0 & 0 & 1 & 16 & 55 \\ 0 & 0 & 0 & 101 & 0 \\ 0 & 0 & 0 & 0 & 101 \end{pmatrix} \qquad \mathbf{B}' = \begin{pmatrix} 0 & -2 & -1 & -2 & -2 \\ 4 & 0 & -1 & 3 & 0 \\ 0 & -3 & 5 & 0 & 1 \\ -3 & -3 & -2 & 0 & 4 \\ 10 & 1 & 0 & -10 & 10 \end{pmatrix}$$

Which is fed to the Babai nearest plane algorithm with input point $\mathbf{As} + \mathbf{e}$. This algorithm will output $\mathbf{As} = \begin{pmatrix} 81 & 58 & 7 & 15 & 88 \end{pmatrix}^T$. If it is multiplied on the left with $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$, the solution is found: $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{As} = \mathbf{s} = \begin{pmatrix} 72 & 46 & 70 \end{pmatrix}^T$ which was indeed the solution of the problem.

4.3.2 BKZ Algorithm.

An algorithm that improves the LLL algorithm is the Block Korkin-Zolotarev (BKZ) algorithm. I use the next section to explain the algorithm and to discuss the properties, following [8] and [20].

Definition 36. Given lattice with basis $\mathbf{B} := \{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ Denote by

$$\pi_i : \mathbb{R}^m \mapsto \text{span}(\mathbf{b_1}, \ldots, \mathbf{b_{i-1}})^\perp$$

the projection onto the space orthogonal to $\text{span}(\mathbf{b_1}, \ldots, \mathbf{b_{i-1}})$, where π_1 is the identity map.

To get an intuitive understanding of the map π_i, refer to the three dimensional example in figure 5.

Proposition 37. The resemblance between the Gram-Schmidt vectors $\mathbf{b_i^*}$ of theorem 21 and the map π_i on the basis vectors is the following:

$$\pi_i(\mathbf{b_j}) = \begin{cases} \mathbf{b_j^*} - \sum_{i \leq \ell < j} \mu_{\ell,i} \mathbf{b}_\ell & \text{if } i \leq j \\ 0 & \text{if } i > j \end{cases}$$

Remark. In particular, $\pi_i(\mathbf{b_i}) = \mathbf{b_i^*}$.

Definition 38. Denote a local block $L_{[i,j]}$ by the projective sublattice

$$L_{[i,j]} := \{ \quad a_i \pi_i(\mathbf{b_i}) + a_{i+1}\pi_i(\mathbf{b_{i+1}}) + \ldots + a_j \pi_i(\mathbf{b_j}) \quad | \quad a_\ell \in \mathbb{Z} \quad \} \subset \mathbb{R}^n$$

Note that in this definition the order of the basis vectors really can change the projective sublattice, as illustrated in Figure 5.

Definition 39 *(BKZ-reduced).* A lattice basis $\mathbf{B} = \{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ is said to be BKZ reduced with blocksize β, or BKZ_β-reduced, if it is LLL-reduced and for each $1 \leq j \leq n$ it has $||\mathbf{b_j^*}|| = \lambda_1(L_{[j,k]})$ with $k = \min(j + \beta - 1, n)$.[3]

Remark. The critirium for being LLL reduced is very similar to BKZ_2 reduced. In some definitions used in other literature, they even coincide. See [47].

The BKZ algorithm that outputs a BKZ-reduced basis is given in pseudocode in algorithm 6. Some more explaination is done in the following paragraph.

This algorithm depends on a block size β, which is constant all the time. Now j denotes the starting point of the current block, $k = \min(j + \beta - 1, n)$ the endpoint of the current block and $h = \min(k + 1, n)$ denotes the endpoint of the next block. The algorithm begins with the LLL-algorithm on the basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$. After that, it iteratively reduces each local block $L_{[j,k]}$ such that the first vector of each block is the shortest in the corresponding projective lattice.

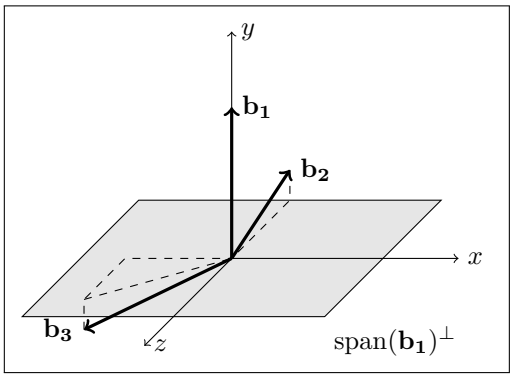
In each iteration, the BKZ algorithm is looking for a vector $\mathbf{v} \in \mathbb{Z}^{k-j+1}$ that satisfies

$$||\pi_j(\sum_{i=j}^{k} v_i \mathbf{b_i})|| = \lambda_1(L_{[j,k]}).$$

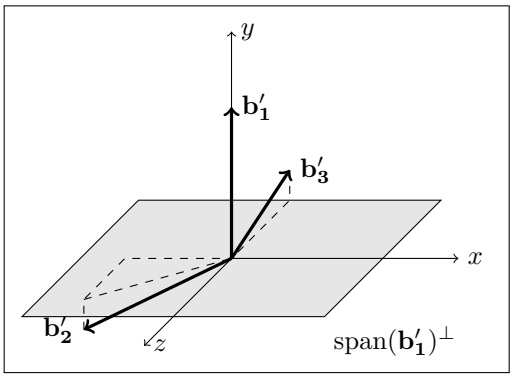
How this vector is found is described in section 4.3.2, for now it is interpreted as a SVP oracle that returns without failure the shortest vector in the projected lattice.

- if $||\mathbf{b_j^*}|| > \lambda_1(L_{[j,k]})$ then a new vector $\mathbf{b} := \sum_{i=j}^{k} v_i \mathbf{b_i}$ is inserted in the basis between $\mathbf{b_{j-1}}$ and $\mathbf{b_j}$. Now the linear dependence, that is added by this vector is removed by calling again the LLL algorithm on this generating set $\{\mathbf{b_1}, \ldots, \mathbf{b_{j-1}}, \mathbf{b}, \mathbf{b_j}, \ldots, \mathbf{b_h}\}$ to obtain an LLL reduced basis. Also μ is updated in accordance to the new LLL-reduced basis. In this case the enumeration was successful.
- If not, apply LLL on the basis $\{\mathbf{b_1}, \ldots, \mathbf{b_h}\}$ and update μ

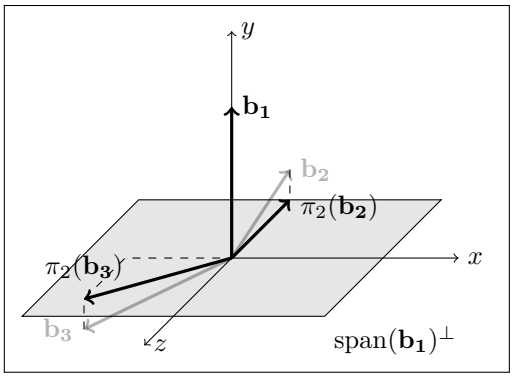
[3] Another common used definition in literature is to replace the latter condition for $\delta||\mathbf{b_j^*}|| \leq \lambda_1(L_{[j,k]})$ for some $0 < \delta < 1$



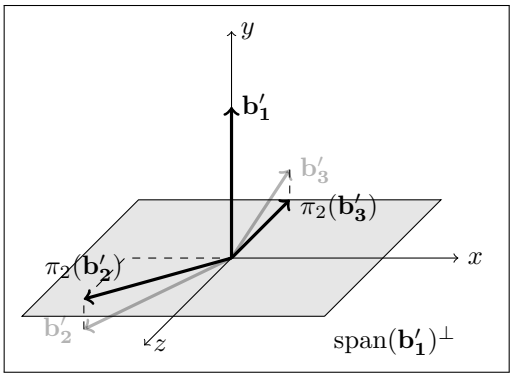
(a) A basis $\{\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}\}$.



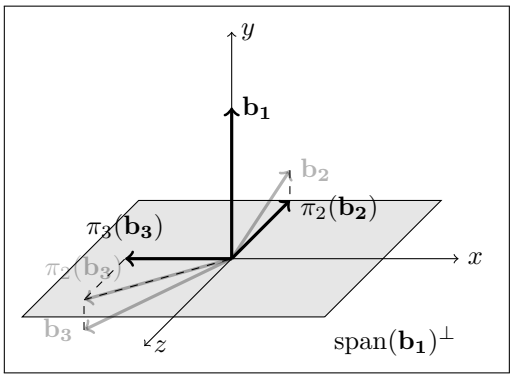
(b) The same basis with $\mathbf{b_2}, \mathbf{b_3}$ swapped.



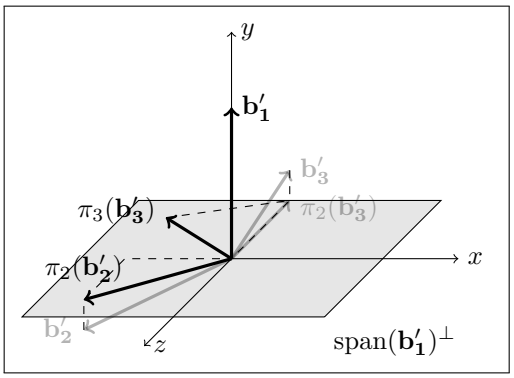
(c) Projection of $\{\mathbf{b_2}, \mathbf{b_3}\}$ under π_2.



(d) Projection $\{\mathbf{b'_2}, \mathbf{b'_3}\}$ under π_2.



(e) Projection of $\mathbf{b_3}$ under π_3.



(f) Projection of $\mathbf{b'_3}$ under π_3.

Figure 5: Projection on the space orthogonal to basis vectors.

Note that the condition $||\mathbf{b_j^*}|| = \lambda_1(L_{[j,k]})$ is equivalent to $\mathbf{v} = (1, 0, \ldots, 0)$, since $\lambda_1(L_{[j,k]}) = ||\pi_j(\sum_{i=j}^{k} v_i \mathbf{b_i})|| = ||\pi_j(\mathbf{b_j})|| = ||\mathbf{b_j^*}||$ At the end of each iteration the basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ is such that $\{\mathbf{b_1}, \ldots, \mathbf{b_h}\}$ is LLL reduced. j runs cyclically through all values of n, unless none of the enumerations were successful. Then the algorithm terminates.

Algorithm 6: BKZ Algorithm

Input: Basis vectors $\mathbf{B} := \{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$, block size β, Gram Schmidt constants $\mu = (\mu_{i,j})$ and $||\mathbf{b_1^*}||, \ldots, ||\mathbf{b_n^*}||$;

$z \leftarrow 0$;
$j \leftarrow 0$;
$\mathbf{B}, \mu \leftarrow \text{LLL}(\mathbf{B}, \mu)$;
while $z < n - 1$ **do**
  $j \leftarrow (j \mod n - 1) + 1$;
  $k \leftarrow \min\{j + \beta + 1, n\}$;
  $h \leftarrow \min\{k + 1, n\}$;
  $\mathbf{v} \leftarrow \text{Enum}(\mu_{j,k}, ||\mathbf{b_j^*}||, \ldots, ||\mathbf{b_k^*}||)$;
  if $\mathbf{v} \neq (1, 0, 0 \ldots, 0)$ **then**
    $z \leftarrow 0$;
    $\{\mathbf{b_1}, \ldots, \mathbf{b_h}\}, \mu \leftarrow \text{LLL}(\{\mathbf{b_1}, \ldots, \mathbf{b_{j-1}}, \sum_{i=j}^{k} v_i \mathbf{b_i}, \mathbf{b_j}, \ldots, \mathbf{b_h}\}, \mu)$;
  end
  else
    $z \leftarrow z + 1$;
    $\{\mathbf{b_1}, \ldots, \mathbf{b_h}\}, \mu \leftarrow \text{LLL}(\mathbf{b_1}, \ldots, \mathbf{b_h}, \mu)$;
  end
end

Output: BKZ$_\beta$-reduced basis $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$;

In each round the BKZ algorithm is looking for the smallest vector in the projected lattice $L_{[j,k]}$. This is a shortest vector problem (SVP) in a lattice of dimension $\leq \beta$.

Enumeration. A way to solve this lower dimension SVP problem is by enumeration, as in [55] and [49].

Let $\{\mathbf{b_1}, \ldots, \mathbf{b_n}\}$ be a basis of a lattice Λ. Suppose one is looking for a shortest vector $\mathbf{x} = x_1\mathbf{b_1} + \ldots + x_n\mathbf{b_n}, x_i \in \mathbb{Z}$ in this lattice. One could do this by enumerating all vectors $\mathbf{y}$ in the lattice inside a ball of radius R: $||\mathbf{y}|| \leq R$, where $R = ||\mathbf{b_1}||$. A shortest vector in this lattice satisfies $||\pi_i(\mathbf{x})|| \leq R$, for all $1 \leq i \leq n$. In particular, $||\pi_n(\mathbf{x})|| \leq R$. From that, conclude

$$\begin{aligned} R &\geq ||\pi_n(\mathbf{x})|| \\ &= ||\pi_n(x_1\mathbf{b_1} + \ldots + x_n\mathbf{b_n})|| \\ &= ||x_1\pi_n(\mathbf{b_1}) + \ldots + x_n\pi_n(\mathbf{b_n})|| \\ &= ||x_n\pi_n(\mathbf{b_n})|| \\ &= |x_n| \cdot ||\mathbf{b_n^*}|| \end{aligned}$$

Hence we found a bound on the absolute value of the n-th coefficient of the shortest vector: $|x_n| \leq \frac{R}{||\mathbf{b_n^*}||}$. This implies we have finite options for choosing our x_n. Now for all possible

values of x_n, we have

$$\begin{aligned}
||\pi_{n-1}(\mathbf{x})|| &= ||\pi_{n-1}(x_1\mathbf{b_1} + \ldots + x_n\mathbf{b_n})|| \\
&= ||x_1\pi_{n-1}(\mathbf{b_1}) + \ldots + x_n\pi_{n-1}(\mathbf{b_n})|| \\
&= ||x_{n-1}\pi_{n-1}(\mathbf{b_{n-1}}) + x_n\pi_{n-1}(\mathbf{b_n})|| \\
&= ||x_{n-1}\mathbf{b^*_{n-1}} + x_n\pi_{n-1}(\mathbf{b_n})|| \\
&= ||x_{n-1}\mathbf{b^*_{n-1}} + x_n(\mathbf{b^*_n} - \mu_{n,n-1}\mathbf{b^*_{n-1}})|| \\
&= ||(x_{n-1} - x_n\mu_{n,n-1})\mathbf{b^*_{n-1}} + x_n\mathbf{b^*_n}|| \\
&\leq R
\end{aligned}$$

From which is concluded

$$\begin{aligned}
|x_{n-1}\mu_{n,n-1} - x_n| \cdot ||\mathbf{b^*_{n-1}}|| &= ||(x_{n-1} - x_n\mu_{n,n-1})\mathbf{b^*_{n-1}}|| \\
&= ||(x_{n-1} - x_n\mu_{n,n-1})\mathbf{b^*_{n-1}} + x_n\mathbf{b^*_n} - x_n\mathbf{b^*_n}|| \\
&\leq ||(x_{n-1} - x_n\mu_{n,n-1})\mathbf{b^*_{n-1}} + x_n\mathbf{b^*_n}|| - ||x_n\mathbf{b^*_n}|| \\
&\leq R - |x_n| \cdot ||\mathbf{b^*_n}||
\end{aligned}$$

From this, conclude that x_{n-1} also lies in a interval of finite length. One can continue this way all the way up to x_1, where each x_i lies in a bounded interval:

$$|x_i + \sum_{j=i+1}^{n} \mu_{j,i}x_j| \cdot ||\mathbf{b^*_i}|| \leq R - \sum_{j=i+1}^{n} |x_j + \sum_{\ell>j} \mu_{\ell,j}x_\ell| \cdot ||\mathbf{b^*_i}||$$

If one wants to find the vector of smallest norm out of this, one can compare the norm of the vectors that are obtained in the way described above. One way to do this is to create an enumeration tree: the root node has exactly one child for each possible value of x_n. On their turn, each child has one child for each possible value of x_{n-1} and so on. At depth n of this search tree, all possible lattice points $\mathbf{y} \in \Lambda$ that have norm $\leq R$. If one naively searches all leaves of this tree for the point with smallest norm, the smallest vector problem is solved. However, this method is not very efficient. In order to reduce the runtime of this subroutine, a technique called pruning is used. The main idea of this technique is that it is most likely for each x_i that it lies in the middle of the finite interval. For the search tree that is created, this implies that the branches that correspond to the extremes of each interval, are "pruned" to avoid unnecessary large growth of the search tree. This is of course at the cost of the risk of losing the desired solution, in case the actual solution was pruned. Further reading on pruning can be found for example in [27].

Sieving. Another technique for solving a shortest vector problem is via sieving. The main idea behind sieving is to start with a list of vectors in the lattice, and combine these vectors to smaller vectors, making the lengths of the vectors in the list iteratively smaller.

More formally, let L be a list of N vectors. Let R be a bounding constant such that $||\mathbf{x}|| \leq R$ for all $\mathbf{x} \in L$ and let γ be a constant < 1. Create a new list L'. Then, for all pairs $\mathbf{x}, \mathbf{y} \in L^2$, $\mathbf{x} \neq \pm\mathbf{y}$, if $||\mathbf{x} \pm \mathbf{y}|| < \gamma R$, add this vector (with the corresponding sign) to the list L'.

Definition 40 *(Independent Identical Distributed vector heuristic)***.** Each time the Sieving algorithm is called, the vectors $\mathbf{x}/||\mathbf{x}||$ for $\mathbf{x}$ the output of the algorithm are i.i.d. uniform distributed points on the unit sphere.

The goal of each iteration is to find vectors of length at most γR. Now for each pair $\mathbf{x}, \mathbf{y} \in L$, assume they are of length approximately R. (This assumption is reasonable, one could immediately add them to L' if they aren't.) For a given $\mathbf{x}$, the condition $||\mathbf{x} - \mathbf{y}|| < \gamma R$ is equal to $\mathbf{y} \in \mathcal{B}(\mathbf{0}, R) \cap \mathcal{B}(\mathbf{x}, \gamma R) =: \mathcal{B}$. If the heuristic from Definition 40 holds, the probability for $\mathbf{y}$ of falling in the region $\mathcal{B}$ is equal to the relative mass of the region $\mathcal{B}$ to the sphere $\mathcal{B}(\mathbf{0}, R)$.

log(Cost estimate)	Model	SVP oracle
0.292β	[14]	Sieving
0.265β	[34]	Sieving
$0.2075\beta + o(\beta)$	[7]	Sieving
$0.3366\beta + 12.31$	[5]	Sieving
$0.187\beta \log \beta - 1.019\beta + 16.1$	[5]	Enumeration
$0.000784\beta^2 + 0.366\beta - 0.9$	[31]	Enumeration

Table 1: Cost estimate models for BKZ_β reduction used in several NIST proposals.

As typically γ is picked close to 1, this probability is $p \approx \sin^n(\pi/3) = (\frac{3}{4})^{n/2}$. In each step we want the list L to not drastically reduce in size. Hence we want to cover the entire sphere $\mathcal{B}(\mathbf{0}, R)$ with balls similar to $\mathcal{B}$ around vector points in the list, thus (assuming the intersections between the spheres are negligible) at least approximately $1/p \approx (\frac{4}{3})^{n/2}$ points in the list are needed to cover the entire sphere. To conclude, if one uses a list of length $\text{poly}(n) \cdot (\frac{4}{3})^{n/2}$, with high probability, a vector can be reduced with at at least one of the other vectors in the list. Thus, in order to cover at least $1 - c$ of the sphere, pick N such that

$$N \cdot \left(\frac{3}{4}\right)^{n/2} \geq 1 - c$$

If $\gamma = 1 - 1/n$ is chosen, it is ensured that only a polynomial number of iterations is needed to go to a list of vectors of norm almost $\lambda_1(\mathcal{L}(\mathbf{A}))$. I conclude that a memory complexity N and time complexity N^2 are necessary for this. [13].

There is currently no closed formula known that describes the cost of BKZ_β reduction. The cost of the BKZ_β algorithm depends on the number of rounds the algorithm runs, and on the cost of finding the smallest vector in each round of the algorithm, via enumeration or sieving (or any other technique). Sieving is asymptotically faster, but enumeration seems to perform quite good in practice for the relatively small β used in practice. [5].

There are many different estimates for the cost of BKZ_β reduction, some more conservative than others. All estimates that authors of NIST proposals adapt in their security analysis are listed in Table 1 with reference. More on these cost models can be found in [3].

Quality estimation of BKZ_β reduction. The next part is used to define some heuristics that later on are used to estimate the quality of BKZ reduced bases.

Definition 41 *(Geometric Series Assumption).* A sufficiently reduced basis has the property that the Gram Schmidt vectors satisfy $||\mathbf{b}^*_{\mathbf{m-i}}|| \approx ||\mathbf{b_m}^*|| \cdot \delta_0^{2i}$. [48, 36]

Definition 42 *(Gaussian Heuristic).* The Gaussian Heuristic is an estimate for the shortest vector in a m-dimensional lattice: [5]

$$\lambda_1(\Lambda) \approx \sqrt{\frac{m}{2\pi e}} \det(\Lambda)^{1/m}$$

In Figure 6, one can see that after a certain point this approximation for the shortest vector isn't accurate anymore. Under the gaussian Heuristic and Geometric assumption, the following holds for BKZ_β reduced lattices [19, 18].

$$\lim_{n \to \infty} \delta_0 \approx \left(\frac{\beta}{2\pi e}(\pi\beta)^{1/\beta}\right)^{\frac{1}{2(\beta-1)}}$$

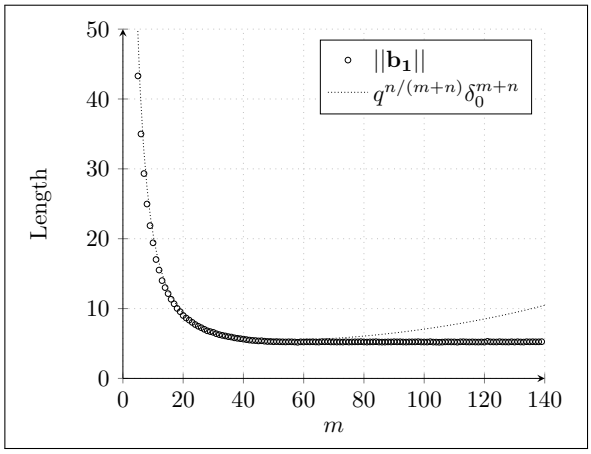
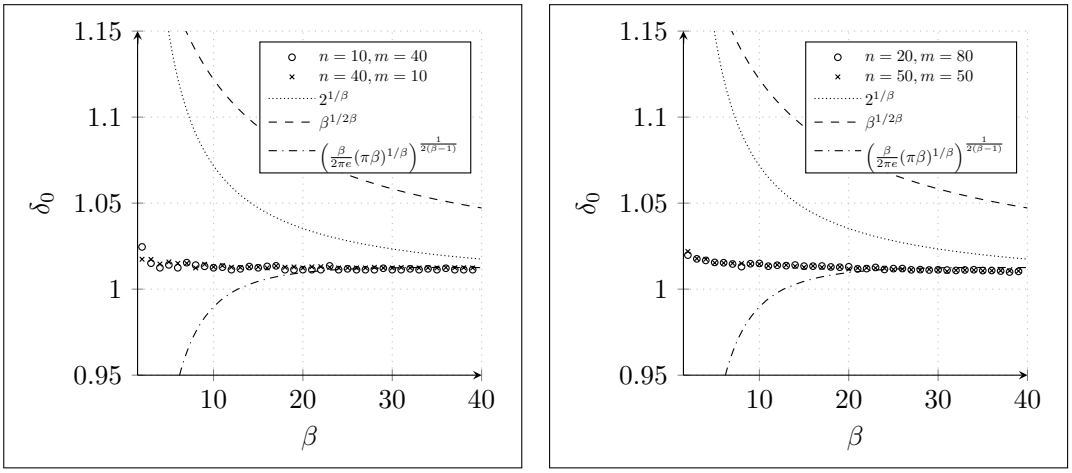


Figure 6: Plot of $q^{n/(n+m)}\delta_0^{m+n}$ against length of $\mathbf{b_1}$ in a m-dimensional dual lattice. $n = 10, q = 251, \beta = 20$



(a) Lattice dimension 50.

(b) Lattice dimension 100.

Figure 7: Comparing the actual Hermite factor to a heuristically chosen.

The latter is also often approximated by $\beta^{1/2\beta}$ or even more simple $2^{1/\beta}$. In Figure 7 one can find how these relate to the actual δ_0 of a random (50,100)-dimensional lattice, that is BKZ_β reduced. A lattice as the primal lattice with $\mathbf{A}^T$ random is considered a random modular lattice. Examples [5, 15], show that this estimation can also be used for finite m.

5 Known attacks against LWE.

In this section I describe three known attacks against the LWE problem. For reference, I describe the cost of exhaustive search. Note that it is known that the primal and dual attack perform quite well for attacking general LWE problems. However, currently there exist more attacks that also achieve good results on the LWE problem. For example attacks based on linearization [10], other algebraic techniques [1] or combinatorial based attacks [4, 33, 29]. For this thesis I focus only on the lattice based primal and dual attack.

5.1 Exhaustive search.

One might wonder what the cost is of just trying all possible secrets, in order to find the correct one. Of course the idea behind this is that it is easy to verify whether a guess is correct or not. Lemma 4 and theorem 1 of [5] are the following:

Lemma 43. Let χ be a gaussian distribution with standard deviation $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$ and mean 0. For a constant $C > 0$ it holds

$$\Pr[e \leftarrow_{\$} \chi : |e| > C\sigma] \leq \frac{2}{C\sqrt{2\pi}}\exp(-\frac{C^2}{2})$$

Proof. For $t > C\sigma$ it holds $\frac{t}{C\sigma} > 1$. Then

$$\begin{aligned}\Pr[e \leftarrow_{\$} \chi : |e| > C\sigma] &= 2\int_{C\sigma}^{\infty} \frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{t^2}{2\sigma^2})dt \\ &= \frac{2}{\sqrt{2\pi}}\int_{C\sigma}^{\infty} \frac{1}{\sigma}\exp(-\frac{t^2}{2\sigma^2})dt \\ &\leq \frac{2}{\sqrt{2\pi}}\int_{C\sigma}^{\infty} \frac{t}{C\sigma^2}\exp(-\frac{t^2}{2\sigma^2})dt \\ &= \frac{2}{C\sqrt{2\pi}}\exp(-\frac{C^2}{2})\end{aligned}$$

which completes the proof. □

Theorem 44. Exhaustive search of the LWE problem with success probability ϵ has time complexity $m \cdot (2t\alpha q + 1)^n \cdot 2n$, memory complexity n and needs $n + m$ samples, where

$$m = \frac{\log(1-\epsilon) - n\log(2t\alpha q + 1)}{\log(2t\alpha)}$$

and $t = 3\sqrt{\log n}$.

Proof. In lemma 1 of [5], one needs n samples to transfer a uniformly random secret LWE problem to a normal LWE problem. By lemma 43, we have that the secret has a high probability to fall in the range $\{-t\alpha q, \ldots, t\alpha q\}$, hence each component of the secret is estimated to fall in this range. Hence $(2t\alpha q + 1)^n$ possible secrets are needed to check to find the correct $\mathbf{s}$. For a correctly guessed $\mathbf{s}$, it holds that $|e_i| = |\mathbf{a_i} \cdot \mathbf{s} - b_i| \leq t\alpha q$ with high probability. For wrong guessed $\mathbf{s}$, this will produce random elements $e_i = \mathbf{a_i} \cdot \mathbf{s} - b_i$ in $\mathbb{Z}_q^n$. A wrong element falls in the range $\{-t\alpha q, \ldots, t\alpha q\}$ with probability $\frac{2t\alpha q+1}{q} \approx 2t\alpha$. In order to accept a wrong guess, e_i has to fall in this range for all samples m. This happens with probability of about $(2t\alpha)^m$. There are $(2t\alpha q + 1)^n$ wrong choices for $\mathbf{s}$. By union bound, the probability of a false positive is $p_f \leq (2t\alpha)^m \cdot (2t\alpha q + 1)^n$. If $p_f = 1 - \epsilon$ is picked, and take $m \geq \frac{\log(p_f) - n\log(2t\alpha+1)}{\log(2t\alpha)}$ samples, the proof is complete.

□

It should be clear that this attack is rather inefficient, even tough it might be optimized to perform better. For example, by starting the exhaustive search with the smallest secret. The next sections describe better methods for solving the LWE problem.

5.2 Primal Attack.

Now try to run the BKZ algorithm (algorithm 6) in combination with the Babai nearest plane method (algorithm 4) with target point $\mathbf{b} = \mathbf{As} + \mathbf{e}$ on the lattice from definition 15:

$$\mathcal{L}(\mathbf{A}) := \{ \quad \mathbf{z} \in \mathbb{Z}^m \quad | \quad \exists \mathbf{s} \in \mathbb{Z}^n; \quad \mathbf{z} = \mathbf{As} \mod q \quad \} \quad \subset \mathbb{Z}^m.$$

The runtime of solving an LWE instance for fixed $n = 20$ with different m is described in figure 8a and 8b, for two values of β. A correct solution of the method is denoted with a dot, while an incorrect solution is denoted with a cross.

Remark. For an LWE problem, it is easy to verify whether an outcome $\mathbf{s}' = \mathbf{s}$ is indeed the correct solution. In this experimental setup, I check this by directly comparing the outcome with the secret key. In practice, an adversary does not have access to this secret as in that case there is no need to do an attack on the cryptosystem.

From these figures we can see that the runtime increases as m increases, which may be no surprise as that is exactly the dimension of the lattice we are attacking. If m is close to n, the attack results in a wrong solution. Choosing m thus has to be done carefully. One could see in Figure 8c and 8c that there is a certain value for m such that if at least m samples are used, the probability of correct decoding is nearly 1.[4]

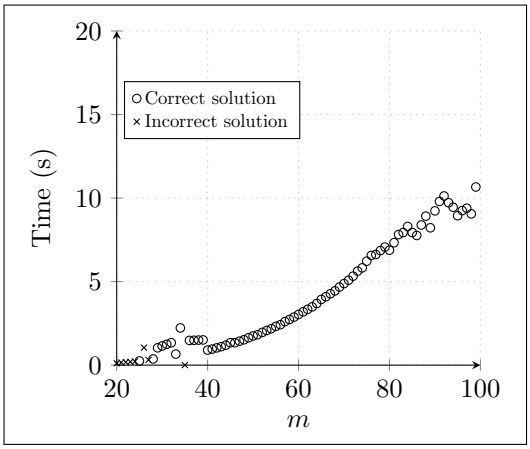
Optimal number of samples. One could try to find a suitable value for m, given the value of n. Take for example the q, σ as in the FrodoKEM specification. Then, to find a suitable m for each n, pick $m = n$, run 100 times the algorithm and if at least $p = 50$ resp. $p = 99$ of the found secrets are correct, accept that as corresponding m value. Otherwise increase m by 1. The corresponding code can be found in Listing 2 and the corresponding results in Figure 12.

```
1  sage: m = 1
2  sage: res = matrix(100,2)
3  sage: for n in range(1,100):
4  ....:     c = 0
5  ....:     esc = 0
6  ....:     print(n)
7  ....:     while c < 100:
8  ....:         if generateAndRecover() == False:
9  ....:             if esc == 0:
10 ....:                 esc = esc +1
11 ....:             else:
12 ....:                 c = 0
13 ....:                 esc = 0
14 ....:                 m = m +1
15 ....:                 print(m)
16 ....:         else:
17 ....:             c = c + 1
18 ....:     res[n,0] = n
19 ....:     res[n,1] = m
```

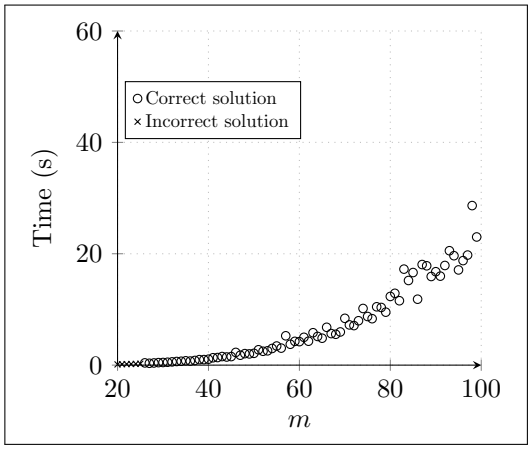
Listing 2: Finding suitable m for a given parameter set q, σ, n

In order to analyze the runtime of recovering the secret vector $\mathbf{s}$, the following experimental setup is used. Start with $n = m = 1$. Then for each $1 \leq n \leq 120$, do the following:

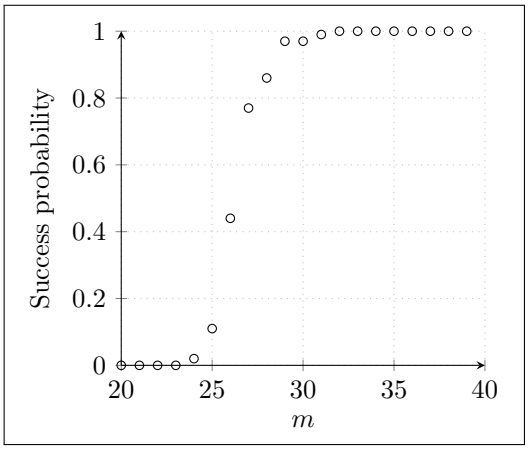
[4] When $\mathbf{A_1}$ is not invertible, this counts as a failure. The matrix can be made nonsingular artificially in several ways. For example by tweaking one value in the matrix and hoping it becomes nonsingular (hoping this doesn't affect the solution) or by replacing a random sample by a new one. As the probability of $\mathbf{A_1}$ being singular is in practice very low, these tricks are omitted.



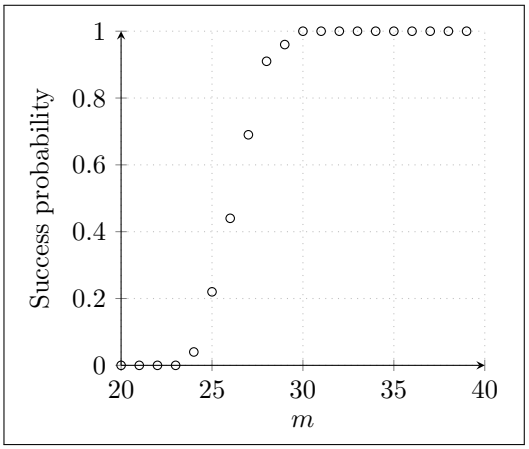
(a) $q = 1009, \sigma = 1, n = 20, \beta = 2$



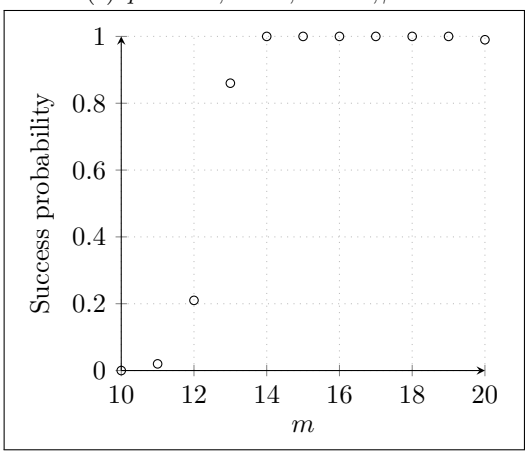
(b) $q = 1009, \sigma = 1, n = 20, \beta = 20$



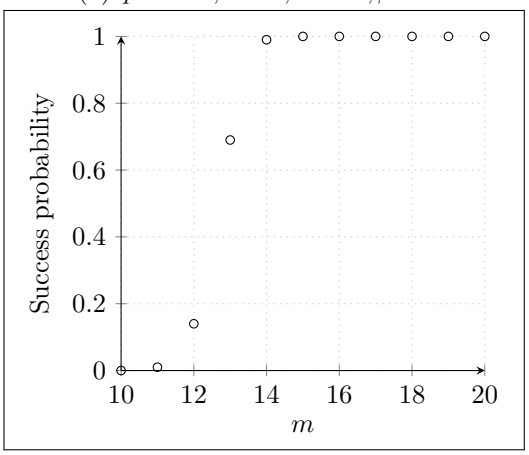
(c) $q = 1009, \sigma = 1, n = 20, \beta = 2$



(d) $q = 1009, \sigma = 1, n = 20, \beta = 20$



(e) $q = 1009, \sigma = 1, n = 10, \beta = 2$



(f) $q = 1009, \sigma = 1, n = 10, \beta = 10$

Figure 8: Solving the LWE problem with different (small scale) parameter sets using the primal attack.

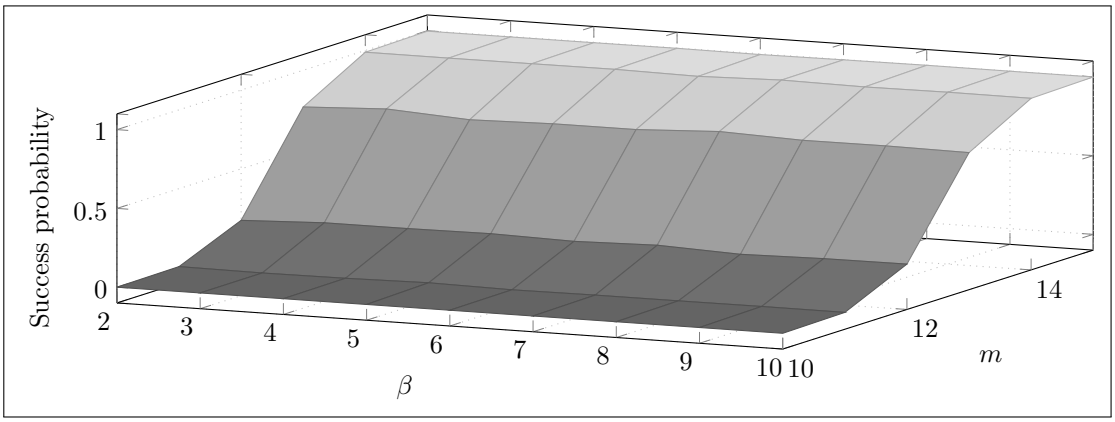


Figure 9: $n = 10, q = 1009, \sigma = 1$

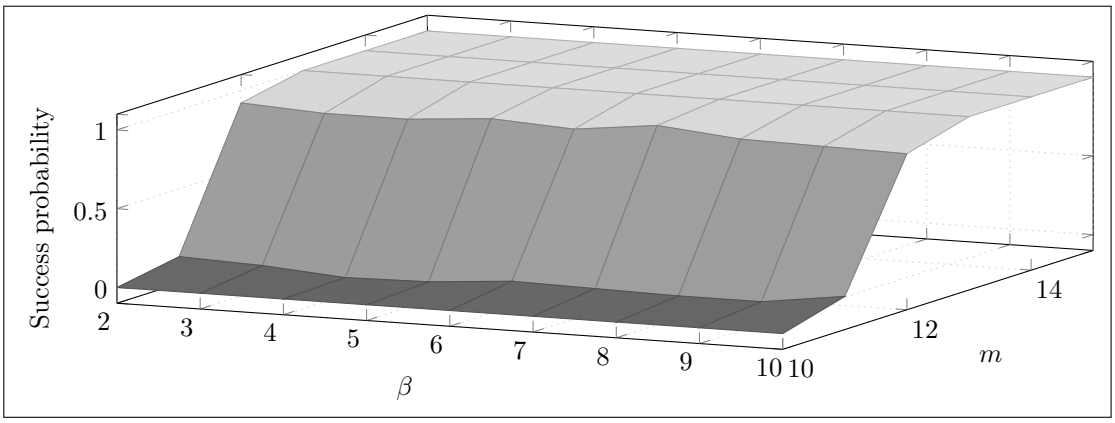


Figure 10: $n = 10, q = 32771, \sigma = 10$

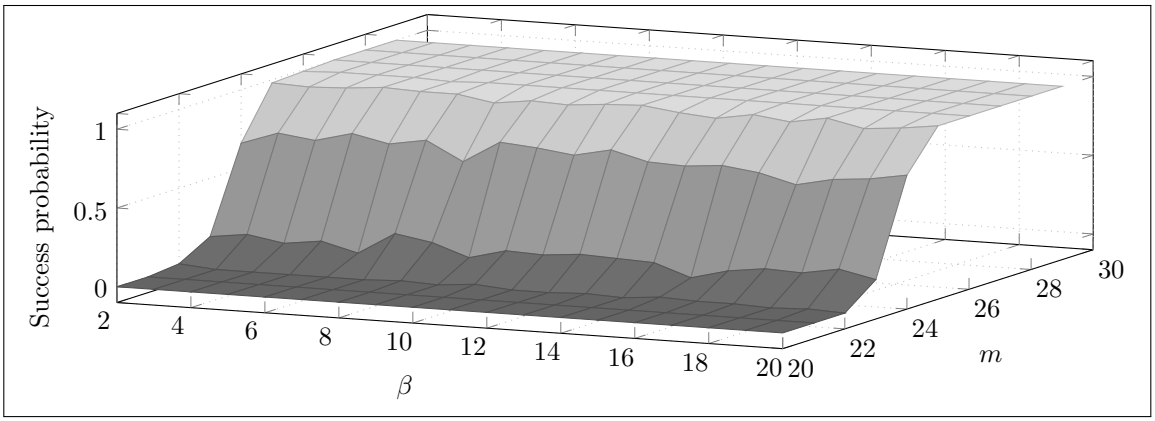
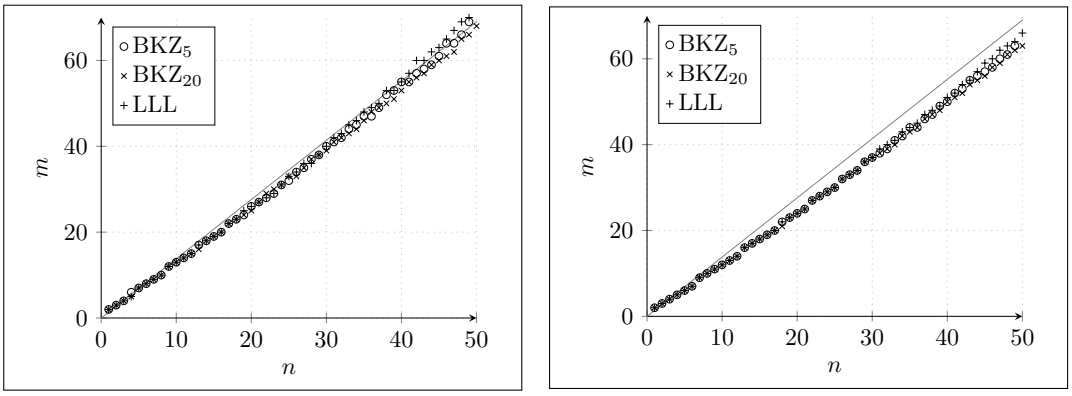


Figure 11: $n = 20, q = 32771, \sigma = 2.8$



(a) $p = 99$ (b) $p = 50$

Figure 12: $q = 32771, \sigma = 2.8$, finding minimal m for different n such that at least p out of 100 attacks are successful. The same line is plotted in both graphs for reference.

(1). Generate m samples divided over two matrices (of n respectively $m - n$ samples each) :

$$\mathbf{A} = \begin{pmatrix} \mathbf{A_1} \\ \mathbf{A_2} \end{pmatrix} \in \mathbb{Z}_q^{m \times n}$$

and generate an m-dimensional error vector $\mathbf{e} \in \mathbb{Z}_q^n$

(2). Generate the basis $\mathbf{B}$ from equation 4 to which one of the lattice reduction algorithms (LLL/BKZ) is applied:

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathbf{A_2}{\mathbf{A_1}}^{-1} & q\mathbf{I}_{m-n} \end{pmatrix} \in \mathbb{Z}_q^{m \times m}$$

(3). Apply a Bounded Distance Decoding Algorithm to the lattice $\mathbf{B}$ with vector $\mathbf{As} + \mathbf{e}$ to find a lattice point $\mathbf{s}'$ and check whether this is the correct solution. (i.e $\mathbf{s}' = \mathbf{s}$ or $\mathbf{As}'$ is distributed according to χ.) If the solution is correct, go to step (5).

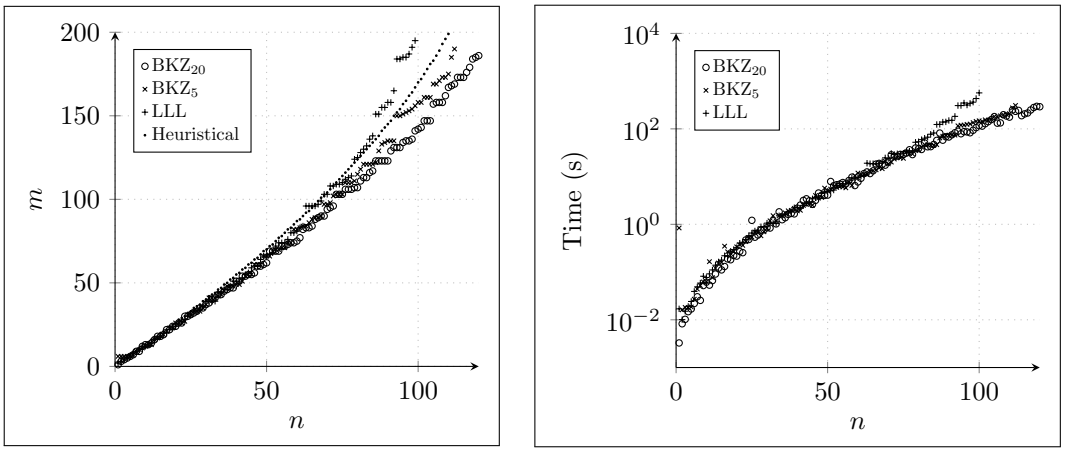
(4). If the solution is incorrect, add one more sample: Generate a new sample $\mathbf{a} \in \mathbb{Z}_q^{1 \times n}$, a new error $e \in Z_q$ according to χ and update

$$\mathbf{A} \leftarrow \begin{pmatrix} \mathbf{A} \\ \mathbf{a} \end{pmatrix} \in \mathbb{Z}_q^{(m+1) \times n}, \mathbf{e} \leftarrow \begin{pmatrix} \mathbf{e} \\ e \end{pmatrix} \in \mathbb{Z}_q^{m+1}, \mathbf{B} \leftarrow \begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathbf{0} \\ \mathbf{A_2}{\mathbf{A_1}}^{-1} & q\mathbf{I}_{m-n} & \mathbf{0} \\ \mathbf{a}\mathbf{A_1^{-1}} & \mathbf{0} & q \end{pmatrix} \in \mathbb{Z}_q^{(m+1) \times (m+1)}$$

Now update $m \leftarrow m + 1$, to have the desired form of the lattice back. Go to step (3).

(5). Update $n \leftarrow n + 1$, start again at step (1).

This procedure can be found in Listing 8. Results of the smallest m for which the procedure succeeds and runtime of the corresponing lattice reduction and decoding can be found in Figure 13. Here LLL, BKZ_5 and BKZ_{20} are used in the reduction step. Later in this section a heuristical method in used to theoretically estimate the required m, which is also plotted in Figure 13a. As one might notice, the minimal required m for correct decoding seems to grow rapidly for fixed β. Actually, for LLL, BKZ_5 and BKZ_{20}, this experiment was not able to find a correct solution for increasing m after $m = 100, m = 112, m = 124$ respectively.



(a) Smallest m for a success.

(b) Runtime of last round.

Figure 13: Solving the LWE problem for $q = 32771, \sigma = 2.8$

Limitations on the number of samples. Despite the fact that in cryptographic settings an adversary wouldn't have access to infinitly many samples, the rest of this paragraph is used to show the limitations of solely increasing the number of samples for solving an LWE problem.

From Claim 23 one concludes that there is correct decoding if $||\mathbf{e}|| \leq \frac{1}{2}\min\{||\mathbf{b_i}^*||\}$. From Lemma 43, it holds with high probability that

$$||\mathbf{e}|| \leq 2\sigma\sqrt{m}. \tag{6}$$

From the Geometric Series assumption, one could conclude that

$$\min\{||\mathbf{b_i}^*||\} = ||\mathbf{b_m}^*|| \approx ||\mathbf{b_1}|| \cdot \delta_0^{-2m}$$

Thus, assuming the Geometric series assumption and the Gaussian Heuristic, one needs to solve

$$||\mathbf{e}|| = \frac{1}{2} q^{(m-n)/m} \delta_0^{-m} \tag{7}$$

If this is combined with the assumption from equation 6, it requires a log root hermite factor that satisfies

$$4\sigma\sqrt{m} \leq q^{1-n/m}\delta_0^{-m}. \tag{8}$$

Which can be solved numerically. In [7] this bound is optimized to

$$\sigma\sqrt{\beta} \leq q^{1-n/m}\delta_0^{2\beta-m-1} \tag{9}$$

In Figure 13a the corresponding required m is plotted against experimentally found m. One can see that actually in practice the algorithm requires a bit fewer samples, due to the optimistic approximation of the length of the error vector.
From equation 9 one can also conclude that for a given parameter set, with fixed β there is a maximum n that it is able to solve. This is made visual in Figure 14.

Remark. In Figure 14 one can see that the maximum is achieved for $m = \sqrt{n \log q / \log \delta_0}$, which we will use later on more often as optimal number of samples.

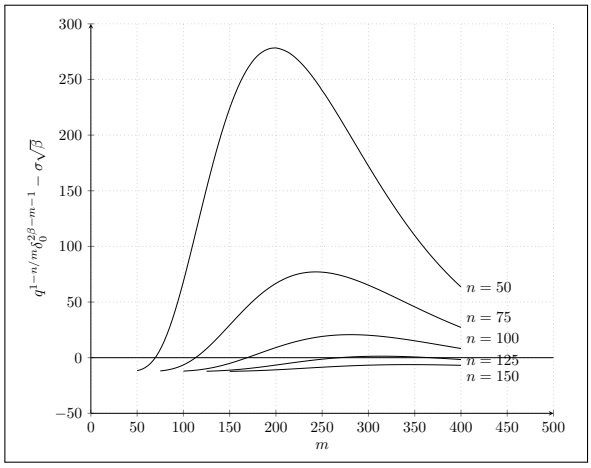


Figure 14: Plot of $q^{1-n/m}\delta_0^{2\beta-m-1} - \sigma\sqrt{\beta}$ for several values of n. $q = 32771, \sigma = 2.8, \beta = 20$. For larger n, this quantity will stay negative hence obtaining no solution in the primal attack.

The primal attack applied to NIST candidates. From [36], assume the optimal number of samples is $m = \sqrt{n \log q / \log \delta_0}$. Thus, for a successful primal attack, find the minimal β that satisfies both:

$$\begin{cases} m & = \sqrt{n \log q / \log \delta_0} \\ \sigma\sqrt{\beta} & \leq q^{1-n/m}\delta_0^{2\beta-m-1} \end{cases} \tag{10}$$

This is done easily numerically, the results of this are shown in Table 2. I chose the most conservative cost function from the point of view of the designer. One can conclude that this primal attack comes quite close to the claimed security of the cryptosystems, except for the Frodo parameters. This is because the authors [6] make some additional conservative assumptions on the developments of the attack in the future.

5.3 Dual Attack.

The Dual attack is a distinguishing attack on the LWE normal form based on a shortest vector problem in the dual lattice $\mathcal{L}^{\perp}(\mathbf{A})$ as in Definition 16:

$$\mathcal{L}^{\perp}(\mathbf{A}) := \{ \quad (\mathbf{y}||\mathbf{z}) \in \mathbb{Z}^n \times \mathbb{Z}^m \quad | \quad \mathbf{y}^T = \mathbf{z}^T\mathbf{A} \mod q \quad \} \quad \subset \mathbb{R}^{m+n}$$

See [2] and [5]. Recall that for a distinguishing attack the adversary is given a tuple $(\mathbf{A}, \mathbf{b})$ either LWE samples or uniformly random. The main strategy is to find a short vector $(\mathbf{y}||\mathbf{z}) \in \mathcal{L}^{\perp}(\mathbf{A})$ and check whether $\mathbf{z} \cdot \mathbf{b}$ is small. This is a useful property, as in the uniformly random case the vector $\mathbf{b}$ is uniformly random, and hence the inner product with $\mathbf{z}$ yields a relatively large value. If $(\mathbf{A}, \mathbf{b})$ are LWE samples, it holds

$$\mathbf{z} \cdot \mathbf{b} = \mathbf{z} \cdot (\mathbf{A}\mathbf{s} + \mathbf{e}) = \mathbf{z} \cdot \mathbf{A}\mathbf{s} + \mathbf{z} \cdot \mathbf{e} = \mathbf{z}^T\mathbf{A}\mathbf{s} + \mathbf{z} \cdot \mathbf{e} = \mathbf{y}^T\mathbf{s} + \mathbf{z} \cdot \mathbf{e} = \mathbf{y} \cdot \mathbf{s} + \mathbf{z} \cdot \mathbf{e}$$

where the rightmost expression is a inner product between vectors that are all small. One concludes that if the error and secret are small enough, one could distinguish between the two cases. The general approach will be

(1). Given m LWE samples as $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, generate

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_n & \mathbf{A}^T \\ \mathbf{0} & \mathbf{I}_m \end{pmatrix}$$

NIST candidate	**Level**	n	q	σ	β	m	$\log_2(\textbf{cost})$	$\log_2(\textbf{claimed security})$
Frodo, [6]	1	640	32768	2.75	480	1379	127.2	103
Frodo, [6]	3	976	65536	2.30	705	2021	186.8	150
NewHope, [7]	1	512	12289	2.00	384	1086	101.8	101
NewHope, [7]	5	1024	12289	2.00	886	2077	234.8	233
KCL-RLWE, [54]	5	1024	12289	2.83	968	2147	256.5	255
BabyBear, [30]	2	624	1024	1.00	577	1188	152.9	152
MamaBear, [30]	5	936	1024	0.94	902	1715	239.0	237
PapaBear, [30]	5	1248	1024	0.87	1219	2220	323.0	320
CRYSTALS-Dilithium, [24]	1	768	8380417	3.74	342	1664	90.6	91
CRYSTALS-Dilithium, [24]	2	1024	8380417	3.15	485	2168	128.5	125
CRYSTALS-Dilithium, [24]	3	1280	8380417	2.00	598	2613	158.5	158
CRYSTALS-Kyber, [16]	1	512	7681	1.58	385	1060	102.0	102
CRYSTALS-Kyber, [16]	3	768	7681	1.41	612	1529	162.2	161
CRYSTALS-Kyber, [16]	5	1024	7681	1.22	829	1974	219.7	218

Table 2: Analysis of the primal attack applied to parameters of NIST candidates with cost function 0.265β. Displayed are the smallest possible β and corresponding number of samples m.

(2). Apply basis reduction BKZ_β or LLL to the lattice generated by $\mathbf{B}$.

(3). By definition the first basis vector $\mathbf{b_1} = (\mathbf{y}||\mathbf{z}) \in \mathbb{Z}^n \times \mathbb{Z}^m$ of the reduced basis is the shortest basis vector.

(4). Calculate $\mathbf{z} \cdot \mathbf{b}$ and decide whether this is small enough.

An adversary has to come up with suitable values for m, β and some kind of boundary value such that $\mathbf{z} \cdot \mathbf{b}$ is accepted as LWE samples.

I start byfinding a suitable number of samples m, which is a delicate matter. Too few samples will yield a lattice that doesn't contain short enough vectors, in case of too many samples the lattice reduction algorithms are not able to find short vectors due to the high dimension [40].

Remark. We are always able to find a vector of length q in the lattice $\mathcal{L}^\perp(\mathbf{A})$, as $(q, 0, 0, \ldots, 0)$ is a vector in this lattice for all $\mathbf{A}$.

From this and the Gaussian Heuristic, conclude that a lattice reduction algorithm is able to find a shortest vector of length approximately $\min\{q, (\det(\Lambda)^{1/(m+n)}\delta_0^{m+n}\} = \min\{q, q^{n/(m+n)}\delta_0^{m+n}\}$. The expression $q^{n/(m+n)}\delta_0^{m+n}$ for m is minimized for optimality. Acquired with differentiation of this expression with respect to m, one concludes that a minimum is achieved when

$$m = \sqrt{\frac{n \log q}{\log \delta_0}} - n. \tag{11}$$

Next, find a suitable value for finding a boundary such that one could conclude that $\mathbf{z} \cdot \mathbf{b}$ is indeed small enough so that we accept it as an LWE sample. Suppose $\mathbf{b} = \mathbf{As} + \mathbf{e}$. Then

$$\mathbf{z} \cdot \mathbf{b} = \mathbf{y} \cdot \mathbf{s} + \mathbf{z} \cdot \mathbf{e} = (\mathbf{y}||\mathbf{z}) \cdot (\mathbf{s}||\mathbf{e})$$

The vector $\mathbf{b_1} = (\mathbf{y}||\mathbf{z})$ has length approximately $q^{n/(m+n)}\delta_0^{m+n}$, thus

$$\sum_{i=1}^{n+m} (b_1)_i \leq q^{n/(m+n)}\delta_0^{m+n}\sqrt{n+m}. \tag{12}$$

and each entry of $\mathbf{c} := (\mathbf{s}||\mathbf{e})$ is from a gaussian distribution χ with standard deviation σ. According to lemma 43 with probability around 95% it holds,

$$|c_i| \leq 2\sigma. \tag{13}$$

Combining equation 12 and 13 gives that with high probability,

$$|\mathbf{z} \cdot \mathbf{b}| = \left| \sum_{i=1}^{n+m} (b_1)_i \cdot c_i \right| \leq \sum_{i=1}^{n+m} |(b_1)_i \cdot c_i| \leq \left| q^{n/(m+n)}\delta_0^{m+n}\sqrt{n+m} \cdot 2\sigma \right|. \tag{14}$$

The next example shows how strong this result is.

Example 45. Given the parameter set $q = 32771, n = 50$. Pick $\beta = 20$ such that one expects after BKZ_β reduction we get a basis with $\delta_0 \approx 1.0133$ (experimentally obtained). For picking the optimal m, pick $m = \sqrt{\frac{n \log q}{\log \delta_0}} - n \approx 150$. Using the result above, one expects to find $|\mathbf{z} \cdot \mathbf{b}| \leq 14904$ for LWE samples. This is not a very strong result, as it is already very close to $\frac{q}{2}$. In Table 3 it can be seen that already for much smaller m one is able to find much smaller values of $|\mathbf{z} \cdot \mathbf{b}|$. This figure shows the average value of $|\mathbf{z} \cdot \mathbf{b}|$ in 100 randomly instantiated LWE problems with the given paramater set of this example.

From [5].

Claim 46. Given LWE instance with parameters n, α, q and vector $\mathbf{v}$. Then the advantage of distinguishing $\mathbf{v} \cdot \mathbf{e}$ from random is approximately $\exp(-\pi||\mathbf{v}||^2\alpha^2)$.

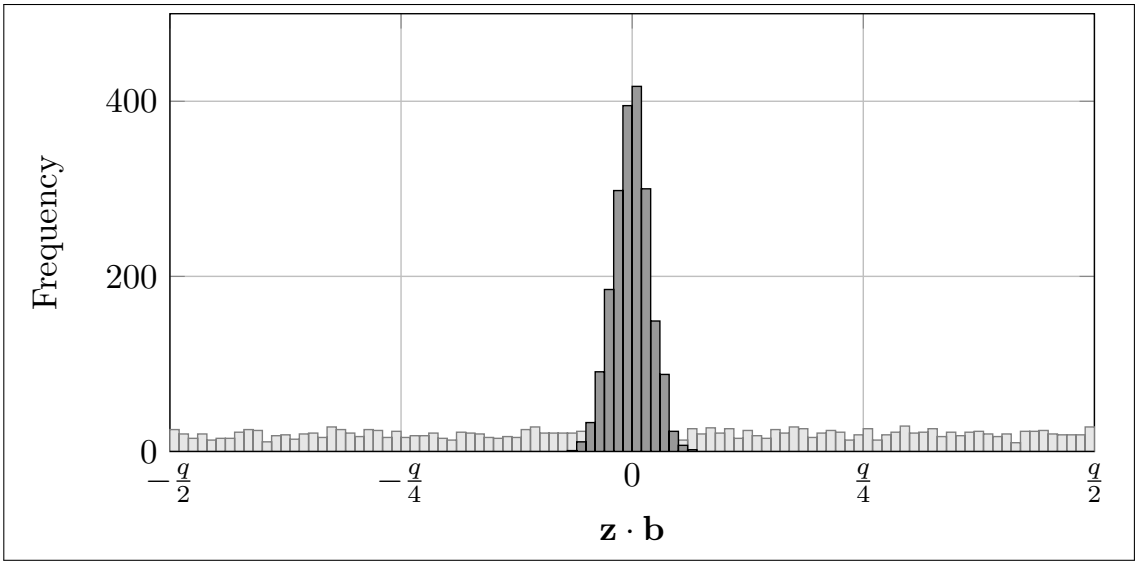


Figure 15: Histogram of output of value $\mathbf{z} \cdot \mathbf{b}$. $q = 32771, \sigma = 2.8, n = 10, m = 10, \beta = 2$. Dark-colored bars are from an LWE distribution, light colored are uniformly random. Each residue class modulo q is denoted with is representative in the interval $(-q/2, q/2]$. 2000 samples were used in this histogram.

m	$\beta = 5$	$\beta = 10$	$\beta = 15$	$\beta = 20$	$\beta = 25$	$\beta = 30$
5	7249	8544	7812	8630	8995	8441
10	9299	8519	7245	7700	9152	7763
15	7587	8891	8167	7198	7474	9029
20	8056	8048	7293	8332	7353	6819
25	7012	6255	6222	6083	5378	5165
30	5533	4358	3992	3507	4161	3346
35	3288	3239	3154	2738	2558	2985
40	2792	2807	2306	2151	2028	1738
45	2029	2007	1906	1825	1778	1538
50	2055	1707	1543	1399	1278	1431
55	1773	1358	1202	1262	1197	926
60	1567	1379	1136	1077	881	808
65	1253	903	898	731	786	632
70	1154	878	890	648	643	564

Table 3: Average value of $|\mathbf{z} \cdot \mathbf{b}|$ for several values of m, β. $n = 50, q = 32771, \sigma = 2.8$

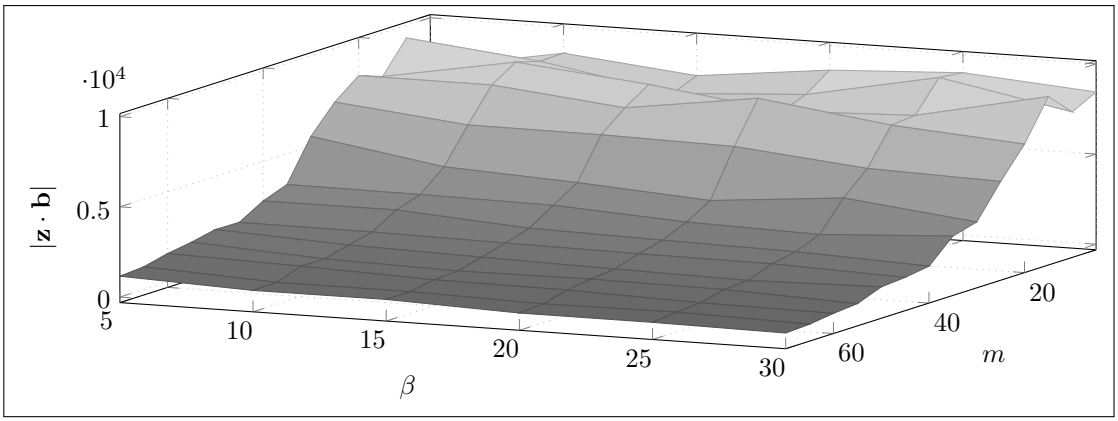


Figure 16: $n = 50, q = 32771, \sigma = 2.8$, the values of Table 3 visually.

m	$\beta = 5$	$\beta = 10$	$\beta = 15$	$\beta = 20$	$\beta = 25$	$\beta = 30$
5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	0.0000	0.0068	0.0018	0.0194	0.1273	0.1599
30	0.1036	0.2833	0.3393	0.4135	0.3135	0.4381
35	0.4470	0.4545	0.4675	0.5311	0.5587	0.4934
40	0.5229	0.5206	0.5972	0.6209	0.6398	0.6841
45	0.6396	0.6430	0.6584	0.6708	0.6780	0.7147
50	0.6356	0.6889	0.7140	0.7360	0.7545	0.7311
55	0.6788	0.7423	0.7661	0.7569	0.7669	0.8083
60	0.7103	0.7390	0.7762	0.7852	0.8152	0.8264
65	0.7583	0.8119	0.8126	0.8382	0.8298	0.8533
70	0.7735	0.8157	0.8139	0.8509	0.8516	0.8637

Table 4: Approximation of the advantage ϵ of the adversary based on the values of Table 3

m	10	20	30	40	50	60	70
$\tilde{\epsilon}$	0.0000	0.0000	0.4135	0.6209	0.7360	0.7852	0.8509
ϵ	0.0000	0.0000	0.0143	0.2294	0.5291	0.7217	0.8262

Table 5: Theoretical advantage ϵ against the approximated advantage $\tilde{\epsilon}$, for $\beta = 20$

Corollary 47. To obtain a success probability of ϵ of solving an LWE instance parametrized by n, α, q using the dual attack, we require a vector $\mathbf{v}$ of norm $||\mathbf{v}|| = \frac{1}{\alpha}\sqrt{\ln \frac{1}{\epsilon}/\pi}$

Proof. Straightforward Calculus. □

Corollary 48. Given an LWE problem with a corresponding dual lattice that is BKZ_β reduced, parametrized by n, α, q and given access to m samples, an adversary has an advantage of

$$\epsilon := \frac{1}{\exp(\pi(\alpha q^{n/(n+m)}\delta_0^{n+m})^2)}$$

in distinghuising LWE samples from uniformly random.

Definition 49 *(Half normal distribution).* Given X that is normal distributed with mean $\mu = 0$ and σ. Then $Y = |X|$ follows the half normal distribution with mean $\mu = \sigma\sqrt{2/\pi}$. Hence a sample from the half normal distribution is within two standard deviations of 0 with probability approximately 95%

Given a parameter set, the data from Figure 15, assuming that $\mathbf{z}\cdot\mathbf{b}$ follows a normal distribution, one can interpret $|\mathbf{z}\cdot\mathbf{b}|$ as a half normal distribution. From that, we approximate the value of $\tilde{\sigma} = |\mathbf{z}\cdot\mathbf{b}| \cdot \sqrt{\pi/2}$. Given such an $\tilde{\sigma}$ for a parameter set, we conclude that for

$$\Pr[|\mathbf{z_0}\cdot\mathbf{b}| < 2\tilde{\sigma}] \approx 0.95$$

Now with this information one can make an distinguishing attack on the LWE problem. Multiply each table entry of Table 3 with $\sqrt{2\pi}$ and use that value as critical value for accepting a pair $(\mathbf{A}, \mathbf{b})$ as LWE samples.
Now the advantage of the adversary, given bound critical value $c < \frac{q}{2}$, is given by $|0.95 - \frac{2c}{q}|$. This translates the average values of Table 3 into the advantage in Table 4.

Lemma 50. An LWE instance parametrized by n, q, α that achieves, using lattice reduction algorithms, a log-root hermite factor

$$\log \delta_0 = \frac{\log^2(\frac{1}{\alpha}\sqrt{\ln \frac{1}{\epsilon}/\pi})}{4n\log q},$$

can be distinguished with advantage ϵ using the dual attack.

Proof. By definition of the hermite factor, it holds $\delta_0^{m+n} = ||\mathbf{b_1}||/q^{n/(n+m)}$. On the other hand, recall from Corollary 47 that in order to get an advantage ϵ one needs to find a vector satisfying $||\mathbf{b_0}|| = \frac{1}{\alpha}\sqrt{\ln \frac{1}{\epsilon}/\pi}$. Combining these gives

$$\delta_0^{m+n} q^{n/(n+m)} = \frac{1}{\alpha}\sqrt{\ln \frac{1}{\epsilon}/\pi}.$$

Now recall from Equation 11 that the optimal number of samples to use is $m = \sqrt{\frac{n\log q}{\log \delta_0}} - n$.

Fill this in and work out to get

$$\begin{aligned}
& & \delta_0^{m+n} q^{n/(m+n)} &= \tfrac{1}{\alpha}\sqrt{\ln \tfrac{1}{\epsilon}/\pi} \\
\Leftrightarrow & & \delta_0^{\sqrt{\frac{n\log q}{\log \delta_0}}} q^{n/(\sqrt{\frac{n\log q}{\log \delta_0}})} &= \tfrac{1}{\alpha}\sqrt{\ln \tfrac{1}{\epsilon}/\pi} \\
\Leftrightarrow & & \log\left(\delta_0^{\sqrt{\frac{n\log q}{\log \delta_0}}} q^{n/(\sqrt{\frac{n\log q}{\log \delta_0}})}\right) &= \log\left(\tfrac{1}{\alpha}\sqrt{\ln \tfrac{1}{\epsilon}/\pi}\right) \\
\Leftrightarrow & & \log\left(\delta_0^{\sqrt{\frac{n\log q}{\log \delta_0}}}\right) + \log\left(q^{n/(\sqrt{\frac{n\log q}{\log \delta_0}})}\right) &= \log\left(\tfrac{1}{\alpha}\sqrt{\ln \tfrac{1}{\epsilon}/\pi}\right) \\
\Leftrightarrow & & \sqrt{\tfrac{n\log q}{\log \delta_0}}\log(\delta_0) + \tfrac{n}{\sqrt{\frac{n\log q}{\log \delta_0}}}\log(q) &= \log\left(\tfrac{1}{\alpha}\sqrt{\ln \tfrac{1}{\epsilon}/\pi}\right) \\
\Leftrightarrow & & \sqrt{n\log(q)}\sqrt{\log(\delta_0)} + \sqrt{\log(\delta_0)}\sqrt{n\log(q)} &= \log\left(\tfrac{1}{\alpha}\sqrt{\ln \tfrac{1}{\epsilon}/\pi}\right) \\
\Leftrightarrow & & 2\sqrt{n\log(q)}\sqrt{\log(\delta_0)} &= \log\left(\tfrac{1}{\alpha}\sqrt{\ln \tfrac{1}{\epsilon}/\pi}\right) \\
\Leftrightarrow & & \sqrt{\log(\delta_0)} &= \tfrac{\log\left(\frac{1}{\alpha}\sqrt{\ln \frac{1}{\epsilon}/\pi}\right)}{2\sqrt{n\log(q)}} \\
\Leftrightarrow & & \log(\delta_0) &= \tfrac{\log^2\left(\frac{1}{\alpha}\sqrt{\ln \frac{1}{\epsilon}/\pi}\right)}{4n\log(q)}
\end{aligned}$$

Which completes the proof. □

Definition 51. Let $C_{n,q,\alpha}(\epsilon)$ denote the log cost of the lattice reduction that is required for achieving advantage ϵ in a dual lattice corresponding to an LWE problem with parameters n, q, α as determined in Lemma 50.

By the Chernoff bound [23], for a distinguishing attack with advantage ϵ, after $1/\epsilon^2$ repetitions, one has a distinguishing attack with a success probability of almost 1.

Now for the cost of solving the LWE problem with the dual attack, note that this dual attack is a distinguishing attack. So repeat this attack nq times as described in Proposition 8 to recover the secret. Conclude that the log cost of the dual attack is

$$\min_{0<\epsilon<1}\left\{C_{n,q,\alpha}(\epsilon) + \log\frac{1}{\epsilon^2} + \log(nq)\right\}$$

Numerically, one could solve this for a parameter set. I applied this to some of the NIST proposals. The results of this can be found in Table 6. As the reader may conclude from this Table, the expected costs of this attack are much higher than the claimed security. This could be explained by the fact that a dual attack can be used to get information about a few elements of the secret and after that apply a primal-like attack on the remaining unknown elements.

NIST candidate	**Level**	n	q	σ	ϵ	β	$\log_2(\textbf{cost})$	$\log_2(\textbf{claimed security})$
Frodo, [6]	1	640	32768	2.75	2^{-12}	585	206.2	103
Frodo, [6]	3	976	65536	2.30	2^{-15}	856	284.6	150
NewHope, [7]	1	512	12289	2.00	2^{-8}	492	171.6	101
NewHope, [7]	5	1024	12289	2.00	2^{-22}	1081	355.9	233
KCL-RLWE, [54]	5	1024	12289	2.83	2^{-24}	1188	388.3	255
BabyBear, [30]	2	624	1024	1.00	2^{-18}	718	247.4	152
MamaBear, [30]	5	936	1024	0.94	2^{-24}	1139	371.8	237
PapaBear, [30]	5	1248	1024	0.87	2^{-26}	1582	494.4	320
CRYSTALS-Dilithium, [24]	1	768	8380417	3.74	2^{-5}	410	153.1	91
CRYSTALS-Dilithium, [24]	2	1024	8380417	3.15	2^{-7}	577	201.8	125
CRYSTALS-Dilithium, [24]	3	1280	8380417	2.00	2^{-8}	710	239.6	158
CRYSTALS-Kyber, [16]	1	512	7681	1.58	2^{-8}	497	172.3	102
CRYSTALS-Kyber, [16]	3	768	7681	1.41	2^{-16}	750	255.1	161
CRYSTALS-Kyber, [16]	5	1024	7681	1.22	2^{-20}	1017	334.3	218

Table 6: Analysis of the dual attack applied to parameters of NIST candidates with cost function 0.265β. Displayed are the smallest possible β and corresponding number of samples m.

6 LWE Optimizations.

In the study of the LWE problem, one finds many different versions of the problem all with some different optimizations. The goal of these optimizations is to make encryption and decryption faster, reduce public key sizes, decrease ciphertext size, reduce memory usage or lower the probability of incorrect decoding without losing too much security. In this section I will discuss some of these optimizations.

6.1 Choice of RWLE modulus.

One could generalize RLWE by allowing more general rings rather than $\mathbb{Z}_q[x]/\langle x^n+1\rangle$. One could take many other polynomials instead of x^n+1. In this section I describe some insecure choices for the polynomial $f(x)$ [17, 25, 43], for which an attack to decision LWE exists.

Proposition 52. Given a polynomial $f(x)$, $\alpha \in \mathbb{Z}_q$ such that $f(\alpha) \equiv 0 \mod q$. Then evaluation at α induces a ring homomorphism

$$\mathrm{ev}_\alpha : R_q \to \mathbb{Z}_q, \qquad g(x) \mapsto g(\alpha).$$

Proof. Let $g \in R_q$. Then clearly $\mathrm{ev}_\alpha(g) \in \mathbb{Z}_q$ as $\alpha \in \mathbb{Z}_q$. Now the only thing that is left to prove is $\mathrm{ev}_\alpha(f(x)) = 0$. But that is true by the assumption that α is a root of f. Hence the evaluation is a homomorphism. □

Suppose $f(\alpha) \equiv 0 \mod q$ for some known α. With proposition 52, consider the following method to solve decision RLWE. For each possible $s \in \mathbb{Z}_q$, calculate the error as $\mathrm{ev}_\alpha(b(x)) - \mathrm{ev}_\alpha(a(x))s$. In case $s = \mathrm{ev}_\alpha(s(x))$, this calculation yields the error term under the evaluation homomorphism $\mathrm{ev}_\alpha(e(x))$ for RLWE samples, which has a small gaussian distribution by assumption. For uniformly random samples this expression is still uniformly random.
To be successful the following requirements are needed:

(1) $f(\alpha) \equiv 0 \mod q$ for some known $\alpha \in \mathbb{Z}_q$.

(2) q shouldn't be too large, we need to go through all of the $s \in \mathbb{Z}_q$ to find the correct one.

(3) One needs to be able to distinguish $\mathrm{ev}_\alpha(e(x))$ from the uniform distribution over $\mathbb{Z}_q$.

Example 53. To demonstrate how powerful this attack is, consider the following example. Let $n = 64$, then $f(x) = x^{64}-1$ is a polynomial satisfying the required conditions. Let $q = 41, \sigma = 1$. Now for each guess $g \in \mathbb{Z}_q$ calculate $b(1) - a(1) \cdot g$ for 10000 samples. Results of these are shown in Figure 17. One can easily verify that this data is uniformly distributed, except for $g = 29$. Note that this is not a very time consuming process, but require quite a lot samples in order to correctly distinguish the two. For greater q, distinguishing the two is easier, but one needs to go through more values to find the correct guess g.

6.2 Small Secret LWE.

Recall that small secret LWE is the version where the secret $\mathbf{s}$ is drawn from a small set. Pick $\{-1, 0, 1\}$, where the probability of sampling a 1 or a -1 equals p, the probability of sampling a 0 is $1 - 2p$ for some $p \in [0, \frac{1}{2}]$.[5]
Each coordinate of $\mathbf{s}$ is from the set $\{-1, 0, 1\}$, where the vector has a fixed Hamming weight h. This version is actually used in a few proposed cryptosystems. For example Lizard [22] or Round5 [11].

[5] This optimization is commonly used in proposed cryptosystems, see for example [3].

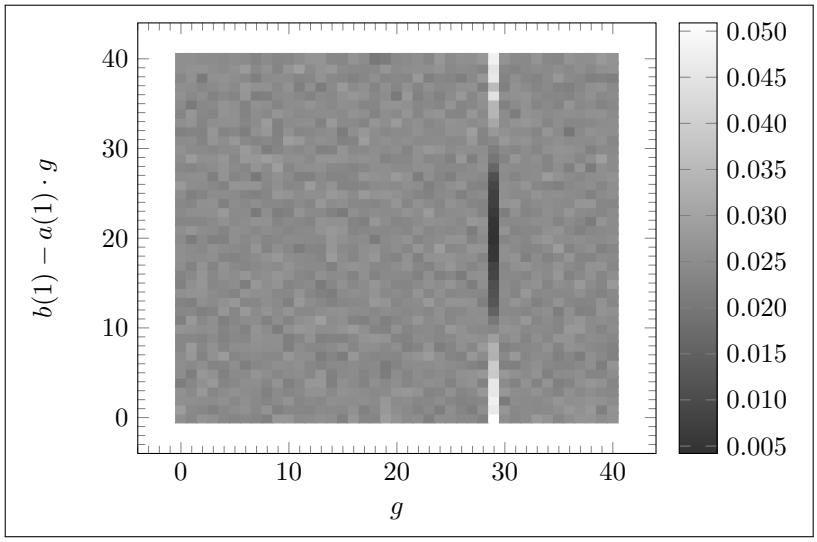


Figure 17: Solving RLWE for an insecure choice of quotient polynomial: $f(x) = x^n - 1$. $q = 41, n = 64, \sigma = 1$. Color indicates the relative occurrence.

6.2.1 Primal attack.

The success of the primal attack relies on the correct decoding of $\mathbf{As} + \mathbf{e}$ to $\mathbf{As}$. Recall from Claim 23 that there is correct decoding if $||\mathbf{e}|| \leq \frac{1}{2}\min\{||\mathbf{b_i^*}||\}$. As the secret distribution has neither effect on the length of the Gram Schmidt basis vectors nor on the error distribution, one could conclude that the smaller secret gives no advantage for an adversary when using the primal attack.

Remark. For n LWE samples, one could interchange the role of the error distribution with the secret distribution: Given n LWE samples in the form $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n$, then $(\mathbf{A}^{-1}, \mathbf{A}^{-1}\mathbf{b}) \in \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n$ are LWE samples where the role of the distributions are interchanged.

The previous remark translates the conditions from equation 10 into the following conditions for correct decoding.

$$\begin{cases} m & = \sqrt{n \log q / \log \delta_0} \\ 2\sqrt{h} & \leq q^{1-n/m}\delta_0^{2\beta-m-1} \end{cases} \tag{15}$$

Again, numerically one could find the smallest β that satisfies both conditions. The result of this is found in Table 7.

However, one could tweak the primal attack in such a way that we may get an advantage [32, 53]. The key idea is to guess k values of the short vector we want to find, and find a shorter vector in a lattice of smaller dimension. Pick $k \in \mathbb{Z}$ such that with high probability $k << h$. Let $\mathbf{A}_\ell \in \mathbb{Z}_q^{(m-n)\times k}$, $\mathbf{A_r} \in \mathbb{Z}_q^{(m-n)\times(n-k)}$ such that $\mathbf{A_2}{\mathbf{A_1}}^{-1} = (\mathbf{A}_\ell||\mathbf{A_r})$. Now define

$$\mathbf{T} := \begin{pmatrix} \mathbf{I}_{n-k} & \mathbf{0}_{(n-k)\times(m-n)} \\ \mathbf{A_r} & q\mathbf{I}_{m-n} \end{pmatrix} \in \mathbb{Z}_q^{(m-k)\times(m-k)}, \quad \mathbf{C} := \begin{pmatrix} \mathbf{0}_{(n-k)\times k} \\ \mathbf{A}_\ell \end{pmatrix} \in \mathbb{Z}_q^{(m-k)\times k}$$

Then the basis of $\mathcal{L}(\mathbf{A})$ can be written as

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}_k & \mathbf{0}_{k\times(m-k)} \\ \mathbf{C} & \mathbf{T} \end{pmatrix}$$

NIST candidate	Level	n	q	h	β	m	$\log_2$(cost)	$\log_2$(claimed security)
Lizard, [22]	1	536	2048	140	449	1056	119.0	130
Lizard, [22]	1	663	1024	128	603	1244	159.8	131
Lizard, [22]	3	816	2048	200	724	1547	191.9	193
Lizard, [22]	3	952	2048	200	847	1772	224.5	195
Lizard, [22]	5	1088	4096	200	899	2023	238.2	257
Lizard, [22]	5	1300	2048	200	1158	2330	306.9	264
nRound2.PKE, [12]	1	442	2659	74	268	818	71.0	74
nRound2.PKE, [12]	2	556	3343	88	346	1014	91.7	97
nRound2.PKE, [12]	3	576	2309	108	385	1046	102.0	106
nRound2.PKE, [12]	4	708	2837	140	485	1273	128.5	138
nRound2.PKE, [12]	5	708	2837	140	485	1273	128.5	138

Table 7: Analysis of the primal attack applied to parameters of NIST candidates with sparse secrets with cost function 0.265β. Displayed are the smallest possible β and corresponding number of samples m.

Now observe that for a small vector $\mathbf{v} = (\mathbf{v_g}||\mathbf{v_l})^T \in \mathcal{L}(\mathbf{A}), \mathbf{v_g} \in \mathbb{Z}_q^k, \mathbf{v_l} \in \mathbb{Z}_q^{m-k}$, it holds that

$$\mathbf{v} = \begin{pmatrix} \mathbf{v_g} \\ \mathbf{v_l} \end{pmatrix} = \mathbf{B} \begin{pmatrix} \mathbf{v_g} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{v_g} \\ \mathbf{C}\mathbf{v_g} + \mathbf{T}\mathbf{x} \end{pmatrix}$$

for some $\mathbf{x} \in \mathbb{Z}^{m-k}$. Now $\mathbf{C}\mathbf{v_g} = -\mathbf{T}\mathbf{x} + \mathbf{v_l}$. This means that $\mathbf{C}\mathbf{v_g}$ lies only a small error vector $\mathbf{v_l}$ away from a lattice point of $\mathcal{L}(\mathbf{T})$. For sufficiently reduced bases, one hopes to find this point via the nearest plane method in the lattice $\mathcal{L}(\mathbf{T})$ on input $\mathbf{C}\mathbf{v_g}$. For a correct guess of $\mathbf{v_g}$, this yields a lattice problem in a $m-k$-dimensional lattice, instead of an m dimensional lattice.

However, the goal of the primal attack is decoding the point $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ to $\mathbf{A}\mathbf{s}$, not finding small vectors in the lattice $\mathcal{L}(\mathbf{A})$. This can be worked around by making a small adjustment in the lattice that is attacked. One knows $\mathbf{b} - \mathbf{e} \in \mathcal{L}(\mathbf{A})$, where $\mathbf{e}$ is the small vector that needs to be found to solve the search LWE problem. Thus $\mathbf{b} - \mathbf{e} = \mathbf{B}\mathbf{x}$ for some $\mathbf{x} \in \mathbb{Z}_q^m$. Define

$$\mathbf{B}' := \begin{pmatrix} 1 & \mathbf{0}_{1\times m} \\ \mathbf{b} & \mathbf{B} \end{pmatrix} \in \mathbb{Z}^{(m+1)\times(m+1)}.$$

Then $(1||\mathbf{e}^T)^T \in \mathcal{L}(\mathbf{B}')$ as

$$\mathbf{B}' \begin{pmatrix} 1 \\ -\mathbf{x} \end{pmatrix} = \begin{pmatrix} 1 \\ \mathbf{b} - \mathbf{B}\mathbf{x} \end{pmatrix} = \begin{pmatrix} 1 \\ \mathbf{e} \end{pmatrix}$$

Next find this vector $(1||\mathbf{e}^T)^T$ using the same approach as before. Let $(1||\mathbf{v_g}^T||\mathbf{v}_\ell{}^T)^T \in \mathcal{L}(\mathbf{B}')$ be the short vector one wants to find. Then (using a suitable subdivision of $\mathbf{b}$),

$$\begin{pmatrix} 1 \\ \mathbf{v_g} \\ \mathbf{v}_\ell \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{0}_{1\times k} & \mathbf{0}_{1\times(m-k)} \\ \mathbf{b_0} & \mathbf{I}_k & \mathbf{0}_{k\times(m-k)} \\ \mathbf{b_1} & \mathbf{C} & \mathbf{T} \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{v_g} - \mathbf{b_0} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} 1 \\ \mathbf{v_g} \\ \mathbf{b_1} + \mathbf{C}(\mathbf{v_g} - \mathbf{b_0}) + \mathbf{T}\mathbf{x} \end{pmatrix}$$

NIST candidate	Level	n	q	h	β	m	k	$\log_2$(cost)	$\log_2$(claimed security)
Lizard, [22]	1	536	2048	140	449	1056	0	119.0	130
Lizard, [22]	1	663	1024	128	589	1244	15	159.5	131
Lizard, [22]	3	816	2048	200	724	1547	0	191.9	193
Lizard, [22]	3	952	2048	200	841	1772	6	224.3	195
Lizard, [22]	5	1088	4096	200	885	2023	16	237.9	257
Lizard, [22]	5	1300	2048	200	1094	2330	72	304.3	264
nRound2.PKE, [12]	1	442	2659	74	316	872	9	85.5	74
nRound2.PKE, [12]	2	556	3343	88	402	1076	12	108.7	97
nRound2.PKE, [12]	3	576	2309	108	453	1112	8	121.8	106
nRound2.PKE, [12]	4	708	2837	140	571	1354	5	152.4	138
nRound2.PKE, [12]	5	708	2837	140	571	1354	5	152.4	138

Table 8: Analysis of the hybrid primal attack applied to parameters of NIST candidates with sparse secrets with cost function 0.265β. Displayed are the smallest possible β, the corresponding number of samples m and the optimal k.

for some $\mathbf{x} \in \mathbb{Z}_q^{m-k}$. Similar as before, now guess $\mathbf{v_g}$ and in case we guessed correctly, apply lattice algorithms on $\mathcal{L}(\mathbf{T})$ to find $\mathbf{v_l}$. Note that $\mathbf{C}(\mathbf{v_g}-\mathbf{b_0})+\mathbf{b_1} = -\mathbf{Tx}+\mathbf{v}_\ell$. And hence is only a small error $\mathbf{v}_\ell$ away from the lattice $\mathcal{L}(\mathbf{T})$. Thus one hopes to find on input $\mathbf{b_1} + \mathbf{C}(\mathbf{v_g} - \mathbf{b_0})$ correctly the error vector $\mathbf{v}_\ell$, when applied to the lattice $\mathcal{L}(\mathbf{T})$ using Babai's nearest plane method.
An implementation of this attack can be found in Appendix A.5. For the consequent guesses of $\mathbf{v_g}$ the values of $f(x)$ are used for ascending x, where

$$f : \{0, \ldots, 3^k - 1\} \to \{0, \pm 1\}^k, \quad x \mapsto \left(\left\lfloor \frac{x}{3^0} \right\rfloor \mod 3, \left\lfloor \frac{x}{3^1} \right\rfloor \mod 3, \ldots, \left\lfloor \frac{x}{3^{k-1}} \right\rfloor \mod 3 \right).$$

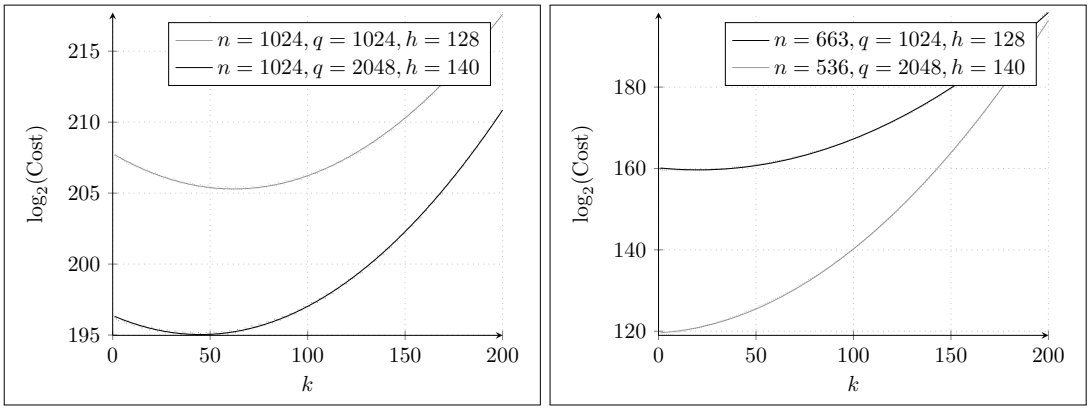
For the modulo 3 calculation representatives $0, \pm 1$ are used.
To analyze the cost of this hybrid primal attack, note that in this hybrid attack one needs to solve a lattice problem in dimension $m - k$. The determinant in unchanged and will remain q^{m-n}. Hence, translate the second condition of 15 into the following (while maintaining the same m).

$$2\sqrt{h} \leq q^{\frac{m-n}{m-k}} \delta_0^{2\beta - m + k - 1} \tag{16}$$

Now the probability that $\mathbf{v_g} = \mathbf{0}_{1\times k}$ is indeed the correct guess, is equal

$$p = \frac{n-h}{n} \cdot \frac{n-h-1}{n} \cdots \frac{n-h-k+1}{n} = \prod_{i=0}^{k-1} \frac{n-h-i}{n} = \frac{(n-h)!}{(n-h-k)!n^k}$$

Thus the expected cost of the attack is the cost of lattice reduction multiplied by $1/p$. For several k, the expected costs are shown in Figure 18. This translates the cost of the hybrid attack as described in Table 8.



(a) Hybrid attack improving primal.

(b) Hybrid attack not improving primal.

Figure 18: Expected cost of the hybrid primal attack for several values of k.

6.2.2 Dual attack.

First let's take a look what happens if one simply ignores the small secret and follows the same approach as described in section 5.3. Recall from equation 14 that with high probability in case of LWE samples (in normal form) it holds that

$$|\mathbf{z} \cdot \mathbf{b}| = \left| \sum_{i=1}^{n+m} (b_1)_i \cdot c_i \right| \leq \sum_{i=1}^{n+m} |(b_1)_i \cdot c_i| \leq \left| q^{n/(m+n)} \delta_0^{m+n} \sqrt{n+m} \cdot 2\sigma \right|.$$

If one changes the secret vector distribution from a Gaussian distribution to the small distribution we have that with high probability a fraction of $\frac{m}{m+n}$ entries are multiplied with an absolute value of at most 2σ. The $\frac{n}{m+n}$ remaining fractions are multiplied with at most ± 1. Therefore when changing from LWE normal form to small secret LWE, one expects to find vectors of length

$$|\mathbf{z} \cdot \mathbf{b}| = \left| \sum_{i=1}^{n+m} (b_1)_i \cdot c_i \right| \leq \sum_{i=1}^{n+m} |(b_1)_i \cdot c_i| \leq \left| q^{n/(m+n)} \delta_0^{m+n} \sqrt{n+m} \cdot (\frac{m}{m+n} 2\sigma + \frac{n}{m+n} 2p) \right|.$$

That is, one finds vectors of a factor $\frac{m+n}{m+\frac{np}{\sigma}}$ smaller, which isn't a huge improvement.

Dimension versus Error tradeoff. Following the reasoning in [2] and [21], with a sparse secret it is reasonable to assume that the last k indices[6] of the secret vector are 0. With this assumption, one is able to parse the public and secret part of our LWE problem in the following way

$$\mathbf{A} = (\mathbf{A_1}||\mathbf{A_2}), \quad \mathbf{A_1} \in \mathbb{Z}_q^{m \times (n-k)}, \quad \mathbf{A_2} \in \mathbb{Z}_q^{m \times k}, \quad \mathbf{s} = (\mathbf{s_1}||\mathbf{s_2}), \quad \mathbf{s_1} \in \mathbb{Z}_q^{n-k}, \quad \mathbf{s_2} \in \mathbb{Z}_q^k.$$

Now the assumption that the last k indices are zero translates to $\mathbf{s_2} = \mathbf{0}_k$. Now for LWE samples it holds that $\mathbf{b} = \mathbf{As} + \mathbf{e} = \mathbf{A_1 s_1} + \mathbf{A_2 s_2} + \mathbf{e} = \mathbf{A_1 s_1} + \mathbf{e}$. Now for the dual attack instead of finding short vectors in $\mathcal{L}^\perp(\mathbf{A})$ one could search for short vectors in $\mathcal{L}^\perp(\mathbf{A_1})$. In other words, one translated the $(m+n)$-dimensional lattice problem to a $(m+n-k)$-dimensional lattice problem. Of course this lattice problem is easier to solve, however, at the cost of a false assumption on $\mathbf{s}$. The probability of a incorrect assumption is $(1-2p)^k$.

[6] An adversary can pick any k indices to be zero, multiplication with a suitable permutation matrix $\mathbf{P}$ gives back the desired form of this attack.

A weaker assumption on the last k indices of $\mathbf{s}$ is to assume that it has at most h nonzero entries. It holds that

$$\mathbf{b} = \mathbf{As} + \mathbf{e} = \mathbf{A_1 s_1} + \mathbf{A_2 s_2} + \mathbf{e}.$$

Now for a short vector $(\mathbf{y_1}, \mathbf{z_1}) \in \mathcal{L}^{\perp}(\mathbf{A_1})$, one has in case of LWE samples

$$\begin{aligned}\mathbf{z_1} \cdot \mathbf{b} &= \mathbf{z_1}^T \mathbf{A_2 s_2} + \mathbf{z_1}^T \mathbf{A_1 s_1} + \mathbf{z_1}^T \mathbf{e} \\ &= \mathbf{z_1}^T \mathbf{A_2 s_2} + \underbrace{\mathbf{y_1} \cdot \mathbf{s_1} + \mathbf{z_1} \cdot \mathbf{e}}_{\text{relatively small}}\end{aligned}$$

Now as $\mathbf{s_2}$ is assumed to be of small weight, one could instead of checking whether $\mathbf{z_1} \cdot \mathbf{b}$ is sufficiently small, check whether $\mathbf{z_1} \cdot \mathbf{b} - \mathbf{z_1} \cdot \mathbf{A_2 s_2}$ is sufficiently small for each possible $\mathbf{s_2}$ of Hamming weight at most h. For this attack, one reduced the dimension of the lattice attack at the cost of losing the correct solution for an incorrect assumption.
Similar to Claim 46, one has the following proposition.

Proposition 54. Given LWE instance with parameters n, q, h and vector $\mathbf{v}$. Then the advantage of distinguishing $\mathbf{v} \cdot \mathbf{e}$ from random is approximately $1 - \frac{2h||\mathbf{v}||}{q\sqrt{m+n}}$.

Proof. Assuming the length of vector $||\mathbf{v}||$ is approximatly equally distributed, i.e. $|v_i| \approx \frac{||\mathbf{v}||}{\sqrt{m+n}}$. Then one expects that $\mathbf{v} \cdot \mathbf{e} = \sum_{i=1}^{m+n} v_i e_i \leq \frac{h||\mathbf{v}||}{\sqrt{m+n}}$ with probability close to 1. For uniformly random samples, we accept them with probability of $\frac{2h||\mathbf{v}||}{\sqrt{m+n}}/q$. Then by definition of the adversary advantage one has

$$\epsilon = 1 - \frac{2h||\mathbf{v}||}{q\sqrt{m+n}}$$

□

Corollary 55. To obtain a success probability of ϵ of solving an LWE instance parametrized by n, α, q using the dual attack, an adversary require a vector $\mathbf{v}$ of norm $||\mathbf{v}|| = \frac{q(1-\epsilon)\sqrt{m+n}}{2h}$

Proof. Straightforward calculus. □

Lemma 56. An LWE instance with sparse ternary secret parametrized by n, q, h that achieves, using lattice reduction algorithms, a log-root hermite factor

$$\log \delta_0 = \frac{\log\left(\frac{q(1-\epsilon)\sqrt{m+n}}{2h}\right) - \frac{n}{m+n}\log q}{m+n},$$

can be distinguished with advantage ϵ using the dual attack with m samples.

Proof. Equivalent to the proof of Lemma 50, where one replaces $\frac{1}{\alpha}\sqrt{\ln \frac{1}{\epsilon}/\pi}$ with $\frac{q(1-\epsilon)\sqrt{m+n}}{2h}$, an adversary needs to apply lattice reduction that satisfy

$$q^{n/(m+n)} \delta_0^{m+n} = \frac{q(1-\epsilon)\sqrt{m+n}}{2h}$$

to find short enough vectors for distinghuising with advantage ϵ. This happens if and only if

$$\begin{aligned} \tfrac{n}{m+n}\log q + (m+n)\log\delta_0 &= \log\left(\tfrac{q(1-\epsilon)\sqrt{m+n}}{2h}\right) \\ \Leftrightarrow \qquad \log\delta_0 &= \tfrac{\log\left(\frac{q(1-\epsilon)\sqrt{m+n}}{2h}\right) - \frac{n}{m+n}\log q}{m+n} \end{aligned}$$

□

Definition 57. Let $C_{n,q,h,m}(\epsilon)$ denote the log cost of the lattice reduction that is required for achieving advantage ϵ in a dual lattice corresponding to sparse secret LWE problem with parameters n, q, h and m samples as determined in Lemma 56.

NIST candidate	Level	n	q	h	ϵ	β	$\log_2$(cost)	$\log_2$(claimed security)
Lizard, [22]	1	536	2048	140	2^{-5}	818	248.2	130
Lizard, [22]	1	663	1024	128	2^{-6}	1189	347.2	131
Lizard, [22]	3	816	2048	200	2^{-6}	1490	428.6	193
Lizard, [22]	3	952	2048	200	2^{-6}	1767	502.5	195
Lizard, [22]	5	1088	4096	200	2^{-6}	1771	504.5	257
Lizard, [22]	5	1300	2048	200	2^{-6}	2486	694.0	264
nRound2.PKE, [12]	1	442	2659	74	2^{-4}	499	161.7	74
nRound2.PKE, [12]	2	556	3343	88	2^{-4}	666	207.2	97
nRound2.PKE, [12]	3	576	2309	108	2^{-5}	795	242.1	106
nRound2.PKE, [12]	4	708	2837	140	2^{-5}	1052	311.0	138
nRound2.PKE, [12]	5	708	2837	140	2^{-5}	1052	311.0	138

Table 9: Analysis of the dual attack applied to parameters of NIST candidates with sparse secrets with cost function 0.265β and $m = 2n$ samples. Displayed are the smallest possible β and the optimal ϵ.

Conclude that the log cost of attacking a sparse secret LWE problem with a dual attack that has m samples is

$$\mathcal{C}_d(n, q, h, m) := \min_{0<\epsilon<1} \left\{ C_{n,q,h,m}(\epsilon) + \log \frac{1}{\epsilon^2} + \log(nq) \right\}.$$

I applied this cost function to NIST proposals, the results can be found in Table 9.

Improvement of hybrid dual attack. The hybrid improvement (of k guesses) of the dual attack searches for a small vector in $\mathcal{L}^\perp(\mathbf{A_1}) \in \mathbb{Z}_q^{m\times(n-k)}$ instead of $\mathcal{L}^\perp(\mathbf{A}) \in \mathbb{Z}_q^{m\times n}$. By proposition 17, this lattice $\mathcal{L}^\perp(\mathbf{A_1})$ has basis

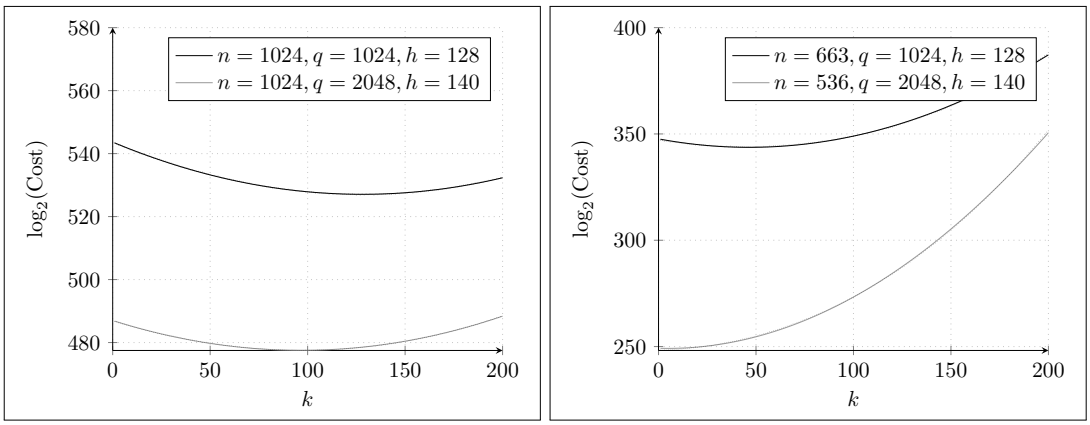
$$\begin{pmatrix} q\mathbf{I}_{n-k} & \mathbf{A_1}^T \\ \mathbf{0}_{m\times(n-k)} & \mathbf{I}_m \end{pmatrix} \in \mathbb{Z}_q^{(m+n-k)\times(m+n-k)}$$

Therefore, the hybrid attack searches for a short vector in a $(m+n-k)-$dimensional lattice that has determinant q^{n-k}. That implies that the cost of the lattice part of a hybrid dual attack with parameter k are $\mathcal{C}(n-k, q, h, m)$. There are 3^k possible different $\mathbf{v_g}$ that needs to be checked, thus the estimated cost of this attack are

$$\mathcal{C}_{h_k}(n, q, h, m, k) := \mathcal{C}_d(n-k, q, h, m) + k \log 3 \tag{17}$$

However, this expected cost can be optimized if an adversary does not try all possible 3^k possibilities, but instead tries only $\mathbf{v_g} = \mathbf{0}_k$. Then as the probability of a correct guess would be $\frac{(n-h)!}{(n-h-k)!n^k}$, the expected cost will be

$$\mathcal{C}_{h_k}(n, q, h, m, k) := \mathcal{C}_d(n-k, q, h, m) + \log\left(\frac{(n-h-k)!n^k}{(n-h)!}\right) \tag{18}$$



(a) Hybrid attack improving dual.

(b) Hybrid attack barely improving dual.

Figure 19: Expected cost of the hybrid dual attack for several values of k.

NIST candidate	**Level**	n	q	h	k	β	$\log_2(\textbf{cost})$	$\log_2(\textbf{claimed security})$
Lizard, [22]	1	536	2048	140	0	817	248.2	130
Lizard, [22]	1	663	1024	128	3	1186	346.7	131
Lizard, [22]	3	816	2048	200	0	1489	428.6	193
Lizard, [22]	3	952	2048	200	1	1764	502.3	195
Lizard, [22]	5	1088	4096	200	0	1770	504.5	257
Lizard, [22]	5	1300	2048	200	21	2448	689.5	264
nRound2.PKE, [12]	1	442	2659	74	2	496	161.7	74
nRound2.PKE, [12]	2	556	3343	88	7	657	206.8	97
nRound2.PKE, [12]	3	576	2309	108	0	793	242.1	106
nRound2.PKE, [12]	4	708	2837	140	0	1050	311.0	138
nRound2.PKE, [12]	5	708	2837	140	0	1050	311.0	138

Table 10: Analysis of the hybrid dual attack applied to parameters of NIST candidates with sparse secrets with cost function 0.265β and $m = 2n$ samples. Displayed are the smallest possible β and optimal k.

7 Sparse Secrets

In this section I discuss the constraints on the value of h when choosing a secure parameter set. I will illustrate how this parameter influences the attack cost of a hybrid primal or dual attack.

For sparse ternary secrets, the hybrid attack is for some parameter sets faster then a regular primal or dual attack. This depends on how sparse the secret is, i.e. how low the value of h is relative to n. When one wants to create a parameter set, it is interesting how sparse a secret can be without losing too much security. With lower h we have smaller secrets and hence a lower probability of decryption failure, but the tradeoff for the guessing in the hybrid attack can be more viable for an adversary. This is the case when the reduction of cost due to the dimensional reduction is less than $1/p$ where p is the probability of a correct guess.

7.1 Primal attack.

In figure 20, one can find the estimated expected cost of the primal attack as a function of k. This is plotted for several different values of h. Observe that for smaller h, the function attains a nontrivial minimum (for larger h the function attains a minimum at $k = 0$). For smaller h, this minimum is attained at larger k, yielding more advantage for an adversary.
A question that naturally arises is how the Hamming weight of the secret is chosen. For the Lizard parameters, the estimated expected cost of the primal attack and the hybrid attack (with corresponding k) are plotted in Figure 21.
One might demand of a parameter set that the hybrid attack won't improve the primal attack. This is the case when the optimal $k = 0$.

Theorem 58. Assume δ_0 is approximated by $2^{1/\beta}$, and assume the logarithm of the cost for BKZ_β reduction is 0.265. Then the hybrid attack improves the primal attack if

$$\frac{\log(\frac{n}{n-h})}{0.265} \leq \frac{m+1}{\frac{m-n}{m}\log q - \log(2\sqrt{h}) + 2} - \frac{m}{\frac{m-n}{m-1}\log q - \log(2\sqrt{h}) + 2} \tag{19}$$

Proof. Note that a hybrid attack that does improve the primal attack (i.e. optimal $k \geq 1$) has expected cost that is is multiplied by $1/p$ where p is the probability of a correct guess. Thus, for a secret with fixed Hamming weight h, one has that for compensating the guess of one entry ($k = 1$) we add a term of $\log(\frac{(n-h-k)!n^k}{(n-h)!}) = \log(\frac{n}{n-h})$ to the estimated logarithm of the cost.
Now assume $\delta_0 = 2^{1/\beta}$. Then, the second line of equation 15 translates into

$$2\sqrt{h} \leq q^{1-n/m}\left(2^{1/\beta}\right)^{2\beta-m-1} = q^{1-n/m}2^{2-\frac{m+1}{\beta}}$$

Hence, for a successful primal attack, pick the smallest β_p that satisfies

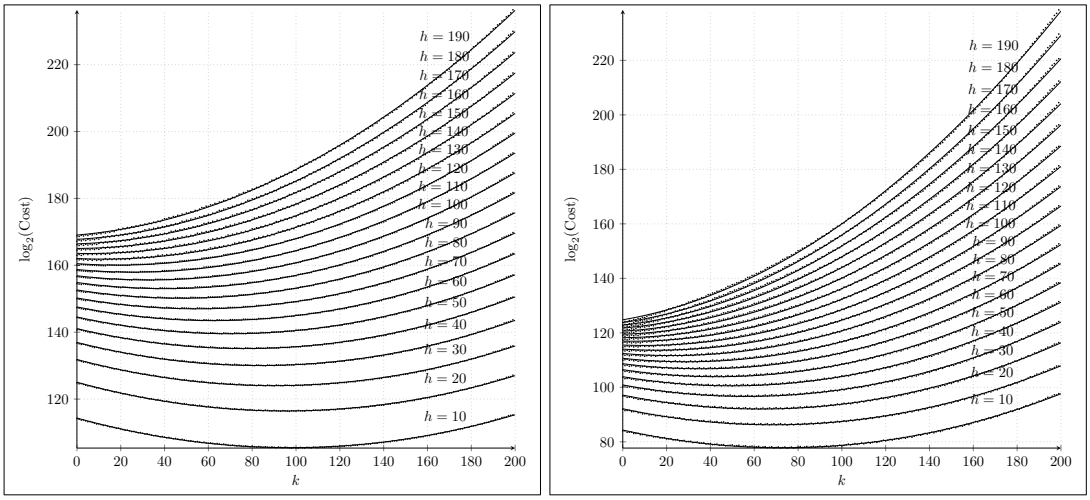
$$\begin{aligned}
& & 2\sqrt{h} &\leq q^{\frac{m-n}{m}}2^{2-\frac{m+1}{\beta_p}} \\
\Leftrightarrow & & \log(2\sqrt{h}) &\leq \tfrac{m-n}{m}\log q + 2 - \tfrac{m+1}{\beta_p} \\
\Leftrightarrow & \log(2\sqrt{h}) - \tfrac{m-n}{m}\log q - 2 &&\leq -\tfrac{m+1}{\beta_p} \\
\Leftrightarrow & \tfrac{m-n}{m}\log q - \log(2\sqrt{h}) + 2 &&\geq \tfrac{m+1}{\beta_p} \\
\Leftrightarrow & & \beta_p &\geq \tfrac{m+1}{\frac{m-n}{m}\log q - \log(2\sqrt{h}) + 2}
\end{aligned}$$

Likewise, for a successful hybrid attack with parameter k, one picks the smallest β_{h_k} that satisfies

$$\beta_{h_k} \geq \frac{m-k+1}{\frac{m-n}{m-k}\log q - \log(2\sqrt{h}) + 2}$$

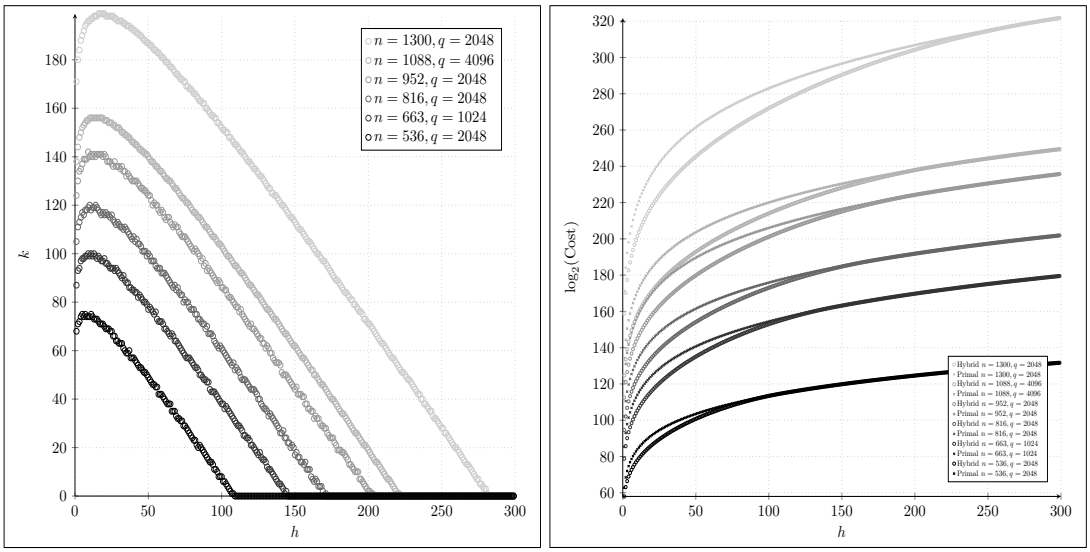
Hence, the improvement in required β due to the dimensional reduction is

$$\beta_p - \beta_{h_k} \approx \frac{m+1}{\frac{m-n}{m}\log q - \log(2\sqrt{h}) + 2} - \frac{m-k+1}{\frac{m-n}{m-k}\log q - \log(2\sqrt{h}) + 2}$$



(a) $n = 663, q = 1024$ (b) $n = 536, q = 2048$

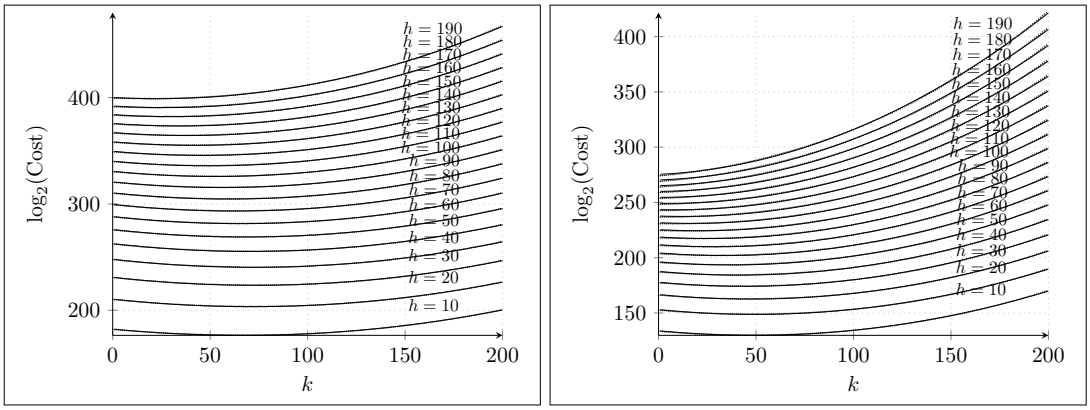
Figure 20: Expected cost of the hybrid primal attack for several values of k, h with Lizard parameters.



(a) Optimal k in the Hybrid primal attack.

(b) Expected cost of the hybrid primal attack for optimal k versus expected cost of the primal attack.

Figure 21: Comparison of primal attack versus hybrid attack as function of h for several parameter sets.



(a) $n = 663, q = 1024$ (b) $n = 536, q = 2048$

Figure 22: Expected cost of the hybrid dual attack for several values of k, h with Lizard parameters.

An adversary multiplying this by 0.265 obtains the reduction in the logarithm of the attack cost. Conclude that the hybrid attack gives improvement if the factor that is added to compensate for the guess is smaller than the estimated cost reduction due to the smaller dimension, and pick $k = 1$. □

7.2 Dual Attack.

Similar to the approach in section 7.1, this section is used to make some statements on how the Hamming weight of a sparse ternary secret affects the estimated cost of an attack.

In figure 22 the cost of the hybrid dual attack are given as function of k. For the parameters of the Lizard [22] encryption scheme, but with different values of h.

Theorem 59. Assume δ_0 is approximated by $2^{1/\beta}$ and assume the logarithm of the cost for BKZ_β reduction is 0.265. Let $0 < \epsilon, \epsilon' < 1$ be the advantages that $C_{n,q,h,m}(\epsilon) - 2\log\epsilon, C_{n-1,q,h,m}(\epsilon') - 2\log\epsilon'$ are minimized, respectively. Then the hybrid attack improves the dual attack if

$$\frac{\log\left(\frac{\epsilon^2(n-1)}{\epsilon'^2(n-h)}\right)}{0.265} \leq \frac{m+n}{\log\left(\frac{q(1-\epsilon)\sqrt{m+n}}{2h}\right) - \frac{n}{m+n}\log q} - \frac{m+n-1}{\log\left(\frac{q(1-\epsilon')\sqrt{m+n-1}}{2h}\right) - \frac{n-1}{m+n-1}\log q} \quad (20)$$

Proof. Note that ϵ, ϵ' that correspond to the minimal choice for $\mathcal{C}_d(n, q, h, m), \mathcal{C}_d(n-1, q, h, m)$. For a dual attack with advantage ϵ an adversary needs a δ_0 that satisfies

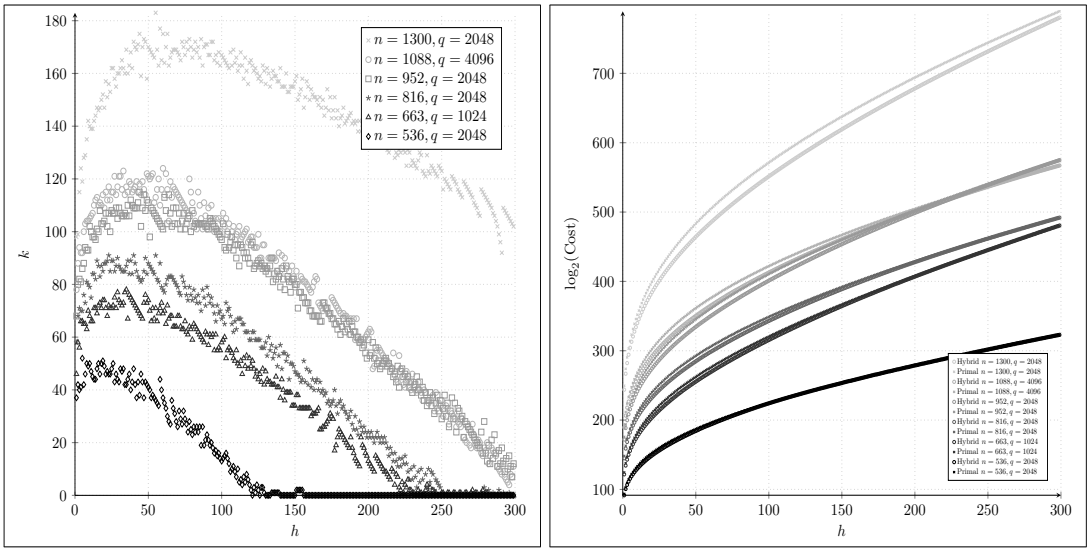
$$\log\delta_0 = \frac{\log\left(\frac{q(1-\epsilon)\sqrt{m+n}}{2h}\right) - \frac{n}{m+n}\log q}{m+n}.$$

With cost function 0.265β and $\delta_0 = 2^{1/\beta}$, the cost of a dual attack are estimated

$$\mathcal{C}_d(n, q, h, m) = \frac{0.265(m+n)}{\log\left(\frac{q(1-\epsilon)\sqrt{m+n}}{2h}\right) - \frac{n}{m+n}\log q} - 2\log\epsilon + \log(nq)$$

Similar for a hybrid attack with $k = 1$ the cost are estimated

$$\mathcal{C}_{h_1}(n, q, h, m) = \frac{0.265(m+n-1)}{\log\left(\frac{q(1-\epsilon')\sqrt{m+n-1}}{2h}\right) - \frac{n-1}{m+n-1}\log q} - 2\log\epsilon' + \log((n-1)q) + \log\left(\frac{n}{n-h}\right)$$



(a) Optimal k in the Hybrid dual attack.

(b) Expected cost of the hybrid attack for optimal k versus expected cost of the dual attack.

Figure 23: Comparison of dual attack versus hybrid attack as function of h for several parameter sets.

There is an improvement if $0 \leq \mathcal{C}_d(n, q, h, m) - \mathcal{C}_{h_1}(n, q, h, m)$, concluding that there is improvement if

$$\frac{\log\left(\frac{\epsilon^2(n-1)}{\epsilon'^2(n-h)}\right)}{0.265} \leq \frac{m+n}{\log\left(\frac{q(1-\epsilon)\sqrt{m+n}}{2h}\right) - \frac{n}{m+n}\log q} - \frac{m+n-1}{\log\left(\frac{q(1-\epsilon')\sqrt{m+n-1}}{2h}\right) - \frac{n-1}{m+n-1}\log q}$$

□

8 Discussion

A designer of a cryptosystem that makes use of sparse secrets should be aware that there exist hybrid attacks that possibly improve a primal or dual attack. It can be of interest for selecting parameter to know for which h there is no improvement anymore. However, a better attack then the primal or dual attack is not a huge problem, as long as the improvement is not too big. If the advantage that the hybrid attack gives, is just a couple of bits security, the system is not immediately broken or weak. A designer that has some specific reason to have very sparse secrets, can for example compensate for this loss by taking a larger dimension.

However, a small side note is needed at the analysis of the hybrid attack. A designer should keep in mind that the cost of the hybrid attack here are the expected estimated cost. It is possible that an adversary, possibly with a really small probability, can make the correct guess and hence be much faster than the expected estimated cost. A designer should also keep in mind that for really sparse secrets, the cost of an exhaustive search may be lower than these hybrid improvements.

My analysis of the LWE problem has a pure mathematical approach. There were no practical constraints imposed on the attacks. For example, an adversary had access to infinitely many samples. In practice this wouldn't be possible. Also, for cryptographic applications, one should keep in mind that the implementation shouldn't allow for side channel attacks.

In conclusion, there is no reason to avoid using sparse ternary secrets in LWE based problems based on the attacks described in this thesis, as long as the designer is aware of the fact that improvements on primal and dual attacks exist. I would also like to stress that the attacks covered in this thesis are not covering the complete package of attacks, as described at the start of section 5. A designer should be aware of more attacks in order to make secure parameter choices, including potential future improvements which are of course hard to predict.

To adapt (R)LWE based cryptographic schemes as the new standard, more research needs to be done. For cryptographic purposes, the RLWE problem has better cryptographic properties than LWE problem, which is at least as hard. It has smaller public key sizes, and polynomial arithmetic can be easily done with this modulus. There is no known attack that exploits the extra structure that is added to achieve these results, provided that a suitable polynomial is chosen. However, that doesn't mean that there exists no such an attack.

Similarly to for example the factoring problem, there is no proof that the LWE problem is indeed a computationally hard problem. Also, there is no proof that a quantum computer cannot break an LWE based cryptosystem (in fact, there even exist quantum improvements of classical attacks, for example [28]). To increase confidence that the problem is hard enough to base cryptographic systems on, the best thing that can be done is more research. There are some questions for LWE problems that are still open: Does there exist an attack that exploits the extra structure for RLWE? Is there any other way to exploit the sparsity of a secret to break the system? Do there exist better attacks against the LWE problem?

If research yields more and better answers to these questions, the confidence in the hardness of the (R)LWE problem is increased, making it more interesting for new cryptographic standards. This doesn't rule out that the post quantum cryptography based on other problems like Multivariate Cryptography, Code-based Cryptography and Supersingular Elliptic Curve Isogeny can turn out to be a better option as a standard for post quantum cryptographic systems. However, at this point in time (R)LWE-based cryptography is a very promising candidate for being the future standard for encryption.

References

[1] Martin Albrecht, Carlos Cid, Jean-Charles Faugere, Robert Fitzpatrick, and Ludovic Perret. Algebraic algorithms for LWE problems, 2014.

[2] Martin R Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 103–129. Springer, 2017.

[3] Martin R Albrecht, Benjamin R Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the {LWE, NTRU} schemes! In *International Conference on Security and Cryptography for Networks*, pages 351–367. Springer, 2018.

[4] Martin R Albrecht, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Lazy modulus switching for the BKW algorithm on LWE. In *International Workshop on Public Key Cryptography*, pages 429–445. Springer, 2014.

[5] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[6] Erdem Alkim, Joppe W Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, Douglas Stebila, et al. Frodokem learning with errors key encapsulation, 2017.

[7] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange—a new hope. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 327–343, 2016.

[8] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 789–819. Springer, 2016.

[9] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Annual International Cryptology Conference*, pages 595–618. Springer, 2009.

[10] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *International Colloquium on Automata, Languages, and Programming*, pages 403–415. Springer, 2011.

[11] Hayo Baan, Sauvik Bhattacharya, Scott Fluhrer, Oscar Garcia-Morchon, Thijs Laarhoven, Ronald Rietman, Markku-Juhani O Saarinen, Ludo Tolhuizen, and Zhenfei Zhang. Round5: Compact and fast post-quantum public-key encryption. In *International Conference on Post-Quantum Cryptography*, pages 83–102. Springer, 2019.

[12] Hayo Baan, Sauvik Bhattacharya, Oscar Garcia-Morchon, Ronald Rietman, Ludo Tolhuizen, Jose Luis Torre-Arce, and Zhenfei Zhang. Round2: KEM and PKE based on GLWR. *IACR Cryptol. ePrint Arch.*, 2017:1183, 2017.

[13] Shi Bai, Thijs Laarhoven, and Damien Stehlé. Tuple lattice sieving. *LMS Journal of Computation and Mathematics*, 19(A):146–162, 2016.

[14] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 10–24. SIAM, 2016.

[15] Nina Bindel, Johannes Buchmann, Florian Göpfert, and Markus Schmidt. Estimation of the hardness of the learning with errors problem with a restricted number of samples. *Journal of Mathematical Cryptology*, 13(1):47–67, 2019.

[16] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[17] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Provably weak instances of Ring-LWE revisited. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 147–167. Springer, 2016.

[18] Hao Chen. A measure version of gaussian heuristic. *IACR Cryptol. ePrint Arch.*, 2016:439, 2016.

[19] Yuanmi Chen. *Réduction de réseau et sécurité concrete du chiffrement completement homomorphe*. PhD thesis, Paris 7, 2013.

[20] Yuanmi Chen and Phong Q Nguyen. BKZ 2.0: Better lattice security estimates. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20. Springer, 2011.

[21] Jung Hee Cheon, Minki Hhan, Seungwan Hong, and Yongha Son. A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret LWE. *IEEE Access*, 7:89497–89506, 2019.

[22] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Lizard: Cut off the tail! a practical post-quantum public-key encryption from LWE and LWR. In *International Conference on Security and Cryptography for Networks*, pages 160–177. Springer, 2018.

[23] Herman Chernoff et al. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[24] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.

[25] Yara Elias, Kristin E Lauter, Ekin Ozman, and E Stange, Katherine. Provably weak instances of Ring-LWE. In *Annual Cryptology Conference*, pages 63–92, 2015.

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

[27] Nicolas Gama, Phong Q Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 257–278. Springer, 2010.

[28] Florian Göpfert, Christine van Vredendaal, and Thomas Wunderer. A hybrid lattice basis reduction and quantum search attack on LWE. In *International Workshop on Post-Quantum Cryptography*, pages 184–202. Springer, 2017.

[29] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In *Annual Cryptology Conference*, pages 23–42. Springer, 2015.

[30] Mike Hamburg. Post-quantum cryptography proposal: ThreeBears. *NIST PQC Round*, 2:4, 2019.

[31] Jeff Hoffstein, Jill Pipher, John M Schanck, Joseph H Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRUEncrypt. In *Cryptographers' Track at the RSA Conference*, pages 3–18. Springer, 2017.

[32] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *Annual International Cryptology Conference*, pages 150–169. Springer, 2007.

[33] Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In *Annual Cryptology Conference*, pages 43–62. Springer, 2015.

[34] Thijs Laarhoven. *Search problems in cryptography.* PhD thesis, PhD thesis, Eindhoven University of Technology, 2015.

[35] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann. 261*, pages 514–534, 1982.

[36] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *Cryptographers' Track at the RSA Conference*, pages 319–339. Springer, 2011.

[37] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer, 2010.

[38] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for Ring-LWE cryptography. Cryptology ePrint Archive, Report 2013/293, 2013. `https://eprint.iacr.org/2013/293`.

[39] Daniele Micciancio and Stephen Checkoway. Lattice algorithms and applications: SVP, CVP and minimum distance. `https://cseweb.ucsd.edu/classes/sp07/cse206a/lec7.pdf`. Visited 13-07-2020.

[40] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. Springer, 2009.

[41] National Insitute of Standards and Technology. Post quantum cryptography standardization process. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`.

[42] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract, 2009.

[43] Chris Peikert. How (not) to instantiate Ring-LWE. In *International Conference on Security and Cryptography for Networks*, pages 411–430. Springer, 2016.

[44] Raymond Puzio and Keenan Kidwell. Proof of Gram Schmidt orthogonalization procedure. `https://planetmath.org/ProofOfGramSchmidtOrthogonalizationProcedure`. Visited 18-05-2020.

[45] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[46] Oded Regev. The learning with errors problem. *Invited survey in CCC*, 7:30, 2010.

[47] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical computer science*, 53(2-3):201–224, 1987.

[48] Claus Peter Schnorr. Lattice reduction by random sampling and birthday methods. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 145–156. Springer, 2003.

[49] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, 1994.

[50] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[51] N. Stephens-Davidowitz. NYU lattices mini course: CVP and babai's algorithm. `http://www.noahsd.com/mini_lattices/05__babai.pdf`. Visited 20-05-2020.

[52] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[53] Thomas Wunderer. Revisiting the hybrid attack: Improved analysis and refined security estimates. *IACR Cryptol. ePrint Arch.*, 2016:733, 2016.

[54] Y Zhao, Z Jin, B Gong, and G Sui. A modular and systematic approach to key establishment and public-key encryption based on lwe and its variants. *NIST PQC Round*, 1:4, 2017.

[55] Zhongxiang Zheng, Xiaoyun Wang, Guangwu Xu, and Yang Yu. Orthogonalized lattice enumeration for solving SVP. *Science China Information Sciences*, 61(3):032115, 2018.

A Code

A.1 Frodo

The following code is an implementation in Sage of the algorithms 1 (`Frodo.Encode`), 2 (`Frodo.Decode`), 9 (`FrodoPKE.KeyGen`), 10 (`FrodoPKE.Enc`) and 11 (`FrodoPKE.Dec`) of the Frodo specification [6]. For simplicity, the algorithms don't use hash functions like AES128 or SHAKE to produce pseudorandom bitstrings, but uses the built-in Sage functions `randint()` and `get_random_element()`. These should suffice for our purposes.

```
D = 15;          # D <= 16
q = 2^D;         #Modulus
n = 8;           #n, m matrix dimensions
m = 8;           #n 0 mod 8
B = 3;           #number of bits encoded in each matrix entry
l = B * n * m #length of the bitstring encoded as m by n matrices
R = Integers(q)
sd = 2.8

D = 10;          # D <= 16
q = 2^D;         #Modulus
n = 1;           #n, m matrix dimensions
m = 1;           #n 0 mod 8
B = 1;           #number of bits encoded in each matrix entry
l = B * n * m #length of the bitstring encoded as m by n matrices
R = Integers(q)
sd = 10


def int2Bbit(x):
    s = [ 0 for i in range(B)]
    if x > 2^B :
        return -1;
    else:
        for i in range(B-1,-1,-1):
            if x >= 2^i:
                x = x - 2^i;
                s[B-i-1] = 1;
    s.reverse()
    return s;

def Bbit2int(b):
    b.reverse()
    x = 0
    for i in range(B):
        x = x + b[i] * 2^(B-i-1);
    return x

def ec(k): return k * 2^(D-B);

def dc(c): return int(Mod(round(int(c)*((2^B) / q)),2^B));

def FrodoEncode(k):
    K = matrix(m,n);
    for i in range(m):
```

```
46          for j in range(n):
47              v = 0
48              for y in range(B):
49                  v = v + k[ (i*n+j)*B+y ]*2^y;
50              ec(v)
51              K[i,j] = ec(v)
52      return K
53
54  def FrodoDecode(K):
55      k = [0 for i in range(l)]
56      for i in range(m):
57          for j in range(n):
58              ks = dc(K[i,j]);
59              kss = int2Bbit(ks);
60              for y in range(B):
61                  k[(i*n+j)*B+y] = kss[y];
62      return k;
63
64  def generateMatrix(n,m,sd):
65          A = matrix(R,n,m)
66          T = RealDistribution('gaussian', sd)
67          for i in range(n):
68                  for j in range(m):
69                          A[i,j] = R(round(T.get_random_element()))
70          return A
71
72  def FrodoGen():
73          A = matrix(n,n)
74          for i in range(n):
75                  for j in range(n):
76                          A[i,j] = R.random_element()
77          S = generateMatrix(n, n, sd)
78          E = generateMatrix(n, n, sd)
79          B = A*S+E
80          for i in range(n):
81                  for j in range(n):
82                          B[i,j] = R(B[i,j])
83          pk = A,B
84          sk = S
85          printASEB(A,S,E,B)
86          return pk, sk;
87
88  def FrodoEnc(mu, pk):
89          A, B = pk
90          Sp = generateMatrix(n, n, sd)
91          Ep = generateMatrix(n, n, sd)
92          Epp = generateMatrix(n, n, sd)
93          Bp = Sp * A + Ep
94          V = Sp * B + Epp
95          C1 = Bp
96          C2 = V + FrodoEncode(mu)
97          printSpEpBpEppV(Sp,Ep,Bp,Epp,V)
98          return C1, C2
```

```
99
100 def FrodoDec(C1, C2 , sk):
101         M = C2–C1*sk
102         printM(M)
103         return FrodoDecode(M)
104
105 def equalBitString(b1,b2):
106     if len(b1) != len(b2):
107         return False
108     for i in range(len(b1)):
109         if b1[i] != b2[i] :
110             return False
111     return True;
112
113 def randomBitString():
114         return [ randint(0,1) for i in range(l)]
115
116 def printASEB(A,S,E,B):
117         print('A')
118         print(A)
119         print('S')
120         print(S)
121         print('E')
122         print(E)
123         print('B')
124         print(B)
125         return
126
127 def printSpEpBpEppV(Sp,Ep,Bp,Epp,V):
128         print('Sp')
129         print(Sp)
130         print('Ep')
131         print(Ep)
132         print('Bp')
133         print(Bp)
134         print('Epp')
135         print(Epp)
136         print('V')
137         print(V)
138         return
139
140 def printM(M):
141         print('M')
142         print(M)
143         return
144
145 pk, sk = FrodoGen()
146 k = randomBitString()
147 K = FrodoEnc(k, pk)
148 k1 = FrodoDec(K[0], K[1], sk)
149 print(equalBitString(k,k1))
```

Listing 3: FrodoPKE.

A.2 LWE.

The following code can be used to instantiate an LWE problem.

```
n = 5
q = 101
R = Integers(q)
sd = 3
m = 7

def generateSecret():
        s = matrix(n,1)
        for i in range(n):
                s[i] = R.random_element()
        return s

def generateSamples():
        A = matrix(R,m,n)
        for i in range(m):
                for j in range(n):
                        A[i,j] = R.random_element()
        return A

def generateError(sd):
        E = matrix(R,m,1)
        T = RealDistribution('gaussian', sd)
        for i in range(m):
                E[i] = R(round(T.get_random_element()))
        return E
```

Listing 4: Basic LWE.

A.3 R-LWE.

The following code can be used to instantiate a RLWE problem.

```
n = 4;
q = next_prime(n)
while mod(q,2*n) != 1:
        q = next_prime(q)
R.<x> = PolynomialRing(Integers(q))
f = R(x^n+1)
Rq.<x> = R.quotient(f)
sigma = 1


def genSecret():
        return Rq.random_element()

def createSample(s):
        a = Rq.random_element()
        T = RealDistribution('gaussian', sigma)
        e = Rq([ round(T.get_random_element()) for i in range(n) ])
        b = a*s+e
        return a,b,e
```

Listing 5: RLWE.

A.4 Primal attack.

```
n = 3
q = next_prime(100)
R = Integers(q)
sd = 2.8
m = 5
beta = 5

def generateSecret():
        s = matrix(n,1)
        for i in range(n):
                s[i] = R.random_element()
        return s

def generateSamples():
        A = matrix(R,m,n)
        for i in range(m):
                for j in range(n):
                        A[i,j] = R.random_element()
        return A

def generateError(sd):
        E = matrix(R,m,1)
        T = RealDistribution('gaussian', sd)
        for i in range(m):
                E[i] = R(round(T.get_random_element()))
        return E

def generateBasis(A):
        A1 = A.delete_rows(range(n,m));
        A2 = A.delete_rows(range(n));
        I = matrix.identity(n);
        zero = matrix(n, m-n);
        if A1.is_singular():
                return -1
        A1inv = A1.inverse();
        qI = q*matrix.identity(m-n);
        B = block_matrix(Integers(),[[I,zero],[A2*A1inv,qI]])
        B.subdivide(None);
        return B.transpose()

def babaiNearestPlane(B,pt):
        w = pt
        B = B.LLL()
        Bg = B.gram_schmidt();
        Bg = Bg[0];

        y = matrix(Integers(),m,m);
        for i in range(m-1,-1,-1):
                l = Bg[i,:]*w/(Bg[i,:]*Bg[i,:].transpose())
                y[i,:] = round(l[0,0]) * B[i,:]
                w = w - (l[0,0]-round(l[0,0]))* Bg[i,:].transpose()-
                    round(l[0,0])*B[i,:].transpose()
```

```
52          v = matrix(Integers(),1,m)
53          for i in range(m):
54                  v = v + y[i,:]
55          return v.transpose();
56
57  def minlength(Bg):
58          mi = sqrt((Bg[0,:]*Bg[0,:].transpose())[0,0])
59          for i in range(m):
60                  c = sqrt((Bg[i,:]*Bg[i,:].transpose())[0,0])
61                  if c < mi:
62                          mi = c
63          return mi.n()
64
65  def sumlength(Bg):
66          sum = 0
67          for i in range(m):
68                  sum = sum + sqrt((Bg[i,:]*Bg[i,:].transpose())[0,0])
69          return sum.n()
70
71  def recoverSecret(A,b,sk):
72          b = matrix(Integers(),b)
73          B = generateBasis(A)
74          if B == -1:
75                  print('Unable to generate Basis')
76                  return -1
77          pt = babaiNearestPlane(B,b)
78          if (A.transpose()*A).is_singular() == True:
79                  print('A^t*A not invertible in recoverSecret')
80                  return -1
81          Ainv = (A.transpose()*A).inverse() * A.transpose()
82          print(Ainv*pt == sk)
83          return Ainv*pt
84
85  def generateExampleBasis():
86          B = matrix([[1,2,3],[3,0,-3],[-3,7,3]])
87          pt = matrix([[10],[6],[5]])
88          return B,pt
89
90  def generateAndRecover():
91          sk = generateSecret();
92          A = generateSamples();
93          if (A.transpose()*A).is_singular() == True:
94                  print('A^t*A not invertible in generateAndRecover')
95                  return -1
96          A1 = A.delete_rows(range(n,m));
97          if A1.is_singular == True:
98                  print('A1 not invertible')
99                  return -1
100         E = generateError(1);
101         pk = (A,A*sk+E);
102         pt = matrix(Integers(),A*sk+E)
103         return sk == recoverSecret(A,pt,sk);
104
```

```
m = 1
res = matrix(50,2)

for n in range ( 40 , 50) :
        c = 0
        esc = 0
        print ( n )
        while c < 100 :
                if generateAndRecover() == False:
                        if esc < 1:
                                esc = esc +1
                                c = c + 1
                        else :
                                c = 0
                                esc = 0
                                m = m +1
                                print (m)
                else :
                        c = c + 1
        res [n , 0 ] = n
        res [n , 1 ] = m
        f = open('primalattackbeta5p99.csv','w')
        f.write(latex(res))
        f.close()
```

Listing 6: Primal attack.

A.5 Hybrid attack

```
n = 20
q = next_prime(2^15)
R = Integers(q)
sd = 1
m = 30
beta = 5
k = 3
p = 0.1

def generateSecret():
        s = matrix(n,1)
        for i in range(n):
                s[i] = R.random_element()
        return s

def generateSamples():
        A = matrix(R,m,n)
        for i in range(m):
                for j in range(n):
                        A[i,j] = R.random_element()
        return A

def generateError(p):
        P = [p, 1-2*p, p]
        X = GeneralDiscreteDistribution(P)
        E = matrix(R,m,1)
        for i in range(m):
                E[i] = X.get_random_element()-1
        return E

def generateBasis(A,b):
        A1 = A.delete_rows(range(n,m));
        A2 = A.delete_rows(range(n));
        I = matrix.identity(n);
        zero = matrix(n, m-n);
        if A1.is_singular():
                return -1,-1
        A1inv = A1.inverse();
        qI = q*matrix.identity(m-n);
        B = block_matrix(Integers(),[[I,zero],[A2*A1inv,qI]])
        B.subdivide(None);
        C = B.delete_rows(range(k))
        C = C.delete_columns(range(k,m))
        T = B.delete_rows(range(k))
        T = T.delete_columns(range(k))
        return C.transpose(), T.transpose()

        A1 = A.delete_rows(range(n,m));
        A2 = A.delete_rows(range(n));
        Ink = matrix.identity(n-k);
        zero1k = matrix(1, k);
        zero1mk = matrix(1,m-k);
```

```
53          zerokmk = matrix(k,m-k)
54
55          if A1.is_singular():
56                  return -1,-1
57          A1inv = A1.inverse();
58          qI = q*matrix.identity(m-n);
59          B = block_matrix(Integers() ,[[I,zero],[A2*A1inv,qI]])
60          B.subdivide(None);
61          return B.transpose()
62
63  def babaiNearestPlane(B,pt):
64          w = pt
65          B = B.BKZ(block_size = beta)
66          Bg = B.gram_schmidt();
67          Bg = Bg[0];
68
69          y = matrix(Integers(),m-k,m-k);
70          for i in range(m-k-1,-1,-1):
71                  l = Bg[i,:]*w/(Bg[i,:]*Bg[i,:].transpose())
72                  y[i,:] = round(l[0,0]) * B[i,:]
73                  w = w - (l[0,0]-round(l[0,0]))* Bg[i,:].transpose()-
                        round(l[0,0])*B[i,:].transpose()
74          v = matrix(Integers(),1,m-k)
75          for i in range(m-k):
76                  v = v + y[i,:]
77          return v.transpose();
78
79  def minlength(Bg):
80          mi = sqrt((Bg[0,:]*Bg[0,:].transpose())[0,0])
81          for i in range(m):
82                  c = sqrt((Bg[i,:]*Bg[i,:].transpose())[0,0])
83                  if c < mi:
84                          mi = c
85          return mi.n()
86
87  def sumlength(Bg):
88          sum = 0
89          for i in range(m):
90                  sum = sum + sqrt((Bg[i,:]*Bg[i,:].transpose())[0,0])
91          return sum.n()
92
93  def recoverSecret(A,b,sk,vg):
94          b = matrix(Integers(),b)
95          C,T = generateBasis(A,b)
96          if C == -1:
97                  print('Unable to generate Basis')
98                  return -1
99          b0 = b.delete_rows(range(k,m));
100         b1 = b.delete_rows(range(k));
101         pt = C.transpose()*(vg-b0)+b1
102         pt = matrix(Integers(),pt)
103         Tx = babaiNearestPlane(T,pt)
104         if (A.transpose()*A).is_singular() == True:
```

```
105                 print('A^t*A not invertible in recoverSecret')
106                 return -1
107         Ainv = (A.transpose()*A).inverse() * A.transpose()
108         ep = block_matrix([[vg],[pt-Tx]])
109         ep.subdivide(None)
110         print(Ainv*(b-ep) == sk)
111         return Ainv*(b-ep)
112
113 def generateExampleBasis():
114         B = matrix([[1,2,3],[3,0,-3],[-3,7,3]])
115         pt = matrix([[10],[6],[5]])
116         return B,pt
117
118 def generateAndRecover2():
119         sk = generateSecret();
120         A = generateSamples();
121         if (A.transpose()*A).is_singular() == True:
122                 print('A^t*A not invertible in generateAndRecover')
123                 return -1
124         A1 = A.delete_rows(range(n,m));
125         if A1.is_singular == True:
126                 print('A1 not invertible')
127                 return -1
128         E = generateError(p);
129         pk = (A,A*sk+E);
130         b = matrix(Integers(),A*sk+E)
131         result = False
132         x = 0
133         print(E)
134         while (result == False) & (x < 3^k):
135                 vg = int2guess(x)
136                 result = (recoverSecret(A,b,sk,vg) == sk)
137                 x = x + 1
138         return result;
139
140 def int2guess(x):
141         vg = matrix(Integers(),k,1)
142         for i in range(k):
143                 vg[i,0] = mod(floor(x/(3^i)),3)
144                 if vg[i,0] == 2:
145                         vg[i,0] = vg[i,0] - 3
146         return vg
147
148 vg = matrix(Integers(),k,1)
149 sk = generateSecret()
150 A = generateSamples()
151 e = generateError(p)
152 b = A*sk+e
153
154
155 def generateAndRecover():
156         vg= matrix(Integers(),k,1)
157         sk = generateSecret();
```

```
158         A = generateSamples();
159         if (A.transpose()*A).is_singular() == True:
160                 print('A^t*A not invertible in generateAndRecover')
161                 return -1
162         A1 = A.delete_rows(range(n,m));
163         if A1.is_singular == True:
164                 print('A1 not invertible')
165                 return -1
166         E = generateError(p);
167         pk = (A,A*sk+E);
168         b = matrix(Integers(),A*sk+E)
169         print(recoverSecret(A,b,sk,vg),E,sk)
170         return sk == recoverSecret(A,b,sk,vg);
```

Listing 7: Hybrid Primal attack.

A.6 Attacking LWE

```
sage: m = 1
sage: for n in range(1,120):
....:     if m < n:
....:         m = n
....:     sk = generateSecret();
....:     A = generateSamples(m);
....:     E = generateError(1);
....:     pk = (A,A*sk+E);
....:     pt = matrix(Integers(),A*sk+E)
....:     B = generateBasis(A)
....:     v1 = walltime()
....:     result = (sk ==recoverSecret(A,pt,sk))
....:     while result  == False:
....:         v2 = walltime()
....:         m = m + 1
....:         ap = generateSamples(1)
....:         A = block_matrix(2,1,[A,ap])
....:         A.subdivide(None);
....:         T = RealDistribution('gaussian', sd)
....:         e = matrix(1)
....:         e[0,0] = R(round(T.get_random_element()))
....:         E = block_matrix(2,1,[E,e])
....:         E.subdivide(None);
....:         pk = (A,A*sk+E);
....:         pt = matrix(Integers(),A*sk+E)
....:         result = (recoverSecret(A,pt,sk) == sk)
....:     res[n,0] = n
....:     res[n,1] = m
....:     res[n,2] = walltime(v1)
....:     res[n,3] = walltime(v2)
....:     f = open('testcsv.csv','w')
....:     f.write(latex(res))
....:     f.close()
```

Listing 8: Attacking the LWE problem

A.7 Estimating the Primal attack.

```
def betatodelta0(beta):
        return (((beta/(2*pi*exp(1)))*(pi*beta)^(1/beta))^(1/(2*beta
            -2))).n()

def minimalbeta(n,q,sigma):
        beta = 40
        delta0 = betatodelta0(beta)
        m = round(sqrt(n*log(q)/log(delta0)).n())
        while( (q^(1-n/m)*delta0^(2*beta-m-1)-sigma*sqrt(beta)).n()
            < 0):
                beta = beta + 1
                delta0 = betatodelta0(beta)
                m = round(sqrt(n*log(q)/log(delta0)).n())
        print('beta = ',beta,', m = ', m)
        print(' & & &', n,'&',q,'&',sigma.n(digits = 3),'&',beta,'&'
            ,m,'&',(beta*0.265).n(digits = 5))
        return beta

def minimalbetah(n,q,h):
        beta = 40
        delta0 = betatodelta0(beta)
        m = round(sqrt(n*log(q)/log(delta0)).n())
        while( (q^(1-n/m)*delta0^(2*beta-m-1)-2*sqrt(h)).n() < 0):
                beta = beta + 1
                delta0 = betatodelta0(beta)
                m = round(sqrt(n*log(q)/log(delta0)).n())
        print('beta = ',beta,', m = ', m)
        print(' & & &', n,'&',q,'&',h,'&',beta,'&',m,'&',(beta
            *0.265).n(digits = 5))
        return beta

def minimalbetahybrid(n,q,h,k):
        beta = 40
        delta0 = betatodelta0(beta)
        m = round(sqrt(n*log(q)/log(delta0)).n())
        while( (q^(1-n/m)*delta0^(2*beta-m-1)-2*sqrt(h)).n() < 0):
                beta = beta + 1
                delta0 = betatodelta0(beta)
                m = round(sqrt(n*log(q)/log(delta0)).n())
        print('Standard: beta = ',beta,', m = ', m,', log(cost) ='
            ,(0.265*beta).n(digits = 5))
        beta = 40
        delta0 = betatodelta0(beta)
        while( (q^((m-n)/(m-k))*delta0^(2*beta-m+k-1)-2*sqrt(h)).n()
            < 0):
                beta = beta + 1
                delta0 = betatodelta0(beta)
        p = factorial(n-h)/(factorial(n-h-k)*n^k)
        print('Hybrid: beta = ',beta,', m = ', m,', log(cost) ='
            ,(0.265*beta+log(1/p)).n(digits = 5))
        return (0.265*beta+log(1/p)).n(digits = 5)

```

```
def hybridcost(n,q,h):
        beta = 40
        delta0 = betatodelta0(beta)
        m = round(sqrt(n*log(q)/log(delta0)).n())
        while( (q^(1-n/m)*delta0^(2*beta-m-1)-2*sqrt(h)).n() < 0):
                beta = beta + 1
                delta0 = betatodelta0(beta)
                m = round(sqrt(n*log(q)/log(delta0)).n())
        mincost = (beta*0.265).n()
        cost = mincost
        k = 1
        mink = 0
        count = 0
        minbeta = beta
        while not((cost > mincost) & (count >10)) :
                beta = 40
                delta0 = betatodelta0(beta)
                while( (q^((m-n)/(m-k))*delta0^(2*beta-m+k-1)-2*sqrt
                    (h)).n() < 0):
                        beta = beta + 1
                        delta0 = betatodelta0(beta)
                p = factorial(n-h)/(factorial(n-h-k)*n^k)
                cost = (0.265*beta+log(1/p)).n()
                if cost < mincost:
                        mincost = cost
                        count = 0
                        mink = k
                        minbeta = beta
                else:
                        count = count + 1
                k = k+1
        print('cost:',mincost,', k =',mink)
        print('&',n,'&',q,'&',h,'&',minbeta,'&',m,'&',mincost,'&')
        return mincost, mink, minbeta,m
```

Listing 9: LWE estimator.