UNIVERSITY OF CALIFORNIA, SAN DIEGO

On the Concrete Security of Lattice-Based Cryptography

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Computer Science

by

Michael Walter

Committee in charge:

Professor Daniele Micciancio, Chair
Professor Mihir Bellare
Professor Shachar Lovett
Professor Hovav Shacham
Professor Kenneth Zeger

2017

The Dissertation of Michael Walter is approved and is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGEMENTS

This work would not have been possible without the support of a large number of people. It is customary, or at least commonly believed to be customary, to thank people in the order of their contribution. However, the people who I am grateful to have supported me in their own different ways and any kind of comparison must fail, so a total ordering is impossible. Instead of attempting the impossible, I will express my gratitude in *some* order but want the reader to know that that order has no special meaning.

I want to start out by thanking my girlfriend, Yvonne Huber, who has stood by me, metaphorically, throughout my time at UCSD, unwavering, despite the years of separation.

I am deeply indebted to my parents and my siblings for their continuous support throughout my entire life. I would most certainly not be here without it.

I am grateful to my advisor and collaborator, Daniele Micciancio, for taking me on as a student and providing me with guidance through all those years, which I can only assume must have taken a great amount of patience on his part.

I thank all my colleagues and friends at UCSD and beyond, but especially Christopher Tosh and Allyson Cauble-Chantrenne, for making the past years such a memorable experience. From intriguing and/or hilarious discussions over lunch breaks to social and recreational activities, without which I would not have lasted for very long: please know, that you all were a part of this.

Finally, I thank all of the staff, faculty, and students at the CSE department of UCSD for making the department what it is: a great place to work. Furthermore, I would like to thank the staff for providing excellent assistance whenever needed.

Chapter 3, in full, is a combination of material as it appears (with minor modifications) in the papers

- "Fast Lattice Point Enumeration with Minimal Overhead" [52] by Daniele Miccian-

cio and Michael Walter, published in the proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016). The dissertation author was the primary investigator and author of this paper.

- "Lattice Point Enumeration on Block Reduced Bases" [79] by Michael Walter, published in the proceedings of the Eighth International Conference on Information Theoretic Security (ICITS 2015). The dissertation author is the sole author of this paper.

- "Practical, Predictable Lattice Basis Reduction" [54] by Daniele Micciancio and Michael Walter, published in the proceedings of the Thirty-Fifth Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2016). The dissertation author was the primary investigator and author of this paper.

Chapter 4, in full, is a reprint of material as it appears (with minor modifications) in the paper "Practical, Predictable Lattice Basis Reduction" [54] by Daniele Micciancio and Michael Walter, published in the proceedings of the Thirty-Fifth Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2016). The dissertation author was the primary investigator and author of this paper.

Chapter 5, in full, is a reprint of material (with minor modifications) that has been submitted for publication and may appear as "On the Bit Security of Cryptographic Primitives" by Daniele Micciancio and Michael Walter. The dissertation author was the primary investigator and author of this paper.

Chapter 6, in full, is a combination of

- material as it appears (with minor modifications) in "Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time" by Daniele Micciancio and Michael Walter, published in the proceedings of the Thirty-Seventh Annual International

Cryptology Conference (CRYPTO 2017). The dissertation author was the primary investigator and author of this paper.

- material (with minor modifications) that has been submitted for publication and may appear as "On the Bit Security of Cryptographic Primitives" by Daniele Micciancio and Michael Walter. The dissertation author was the primary investigator and author of this paper.

Chapter 7, in full, is a reprint of material as it appears (with minor modifications) in "Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time" by Daniele Micciancio and Michael Walter, published in the proceedings of the Thirty-Seventh Annual International Cryptology Conference (CRYPTO 2017). The dissertation author was the primary investigator and author of this paper.

VITA

2009	Bachelor of Science in Computer Science, TU Darmstadt, Germany
2012	Master of Science in Computer Science, TU Darmstadt, Germany
2017	Doctor of Philosophy, University of California, San Diego

ABSTRACT OF THE DISSERTATION

On the Concrete Security of Lattice-Based Cryptography

by

Michael Walter

Doctor of Philosophy in Computer Science

University of California, San Diego, 2017

Professor Daniele Micciancio, Chair

Lattice-based cryptography is an extraordinarily popular subfield of cryptography. But since it is also a very young field, practical proposals for lattice-based cryptographic primitives have only recently started to emerge. Turning a cryptographic scheme into an implementation poses a range of questions, the arguably most important one being its concrete security: how do we ensure that any practically conceivable adversary is unable to break the scheme? In this thesis, we address two issues that arise in this context.

Part I is concerned with basing cryptanalytic tools on a sound theoretical foundation. The common approach to analyzing a concrete cryptographic primitive is to analyze

the performance of known algorithms to estimate the attack complexity of a hypothetical adversary. This requires a thorough theoretical understanding of the best performing algorithms. Unfortunately, for many subclasses of lattice algorithms there is a gap in our understanding, which leads to problems in the cryptanalytic process. In this part of the thesis we address these issues in two closely related subclasses of such algorithms. We develop new algorithms and analyze existing ones and show that in both cases it is possible to obtain algorithms that are simultaneously well understood in theory and competitive in practice.

In Part II we focus on an integral part of most lattice-based schemes: sampling from a specific distribution over the integers. Implementing such a sampler securely and efficiently can be challenging for distributions commonly used in lattice-based schemes. We introduce new tools and security proofs that reduce the precision requirements for samplers, allowing more efficient implementations in a wide range of settings while maintaining high levels of security. Finally, we propose a new sampling algorithms with a unique set of properties desirable for implementations of cryptographic primitives.

Chapter 1

Introduction

Lattice-based cryptography has seen a huge rise in popularity in the last two decades. This is commonly attributed to, among other advantages, its conjectured resistance to quantum computers. It is a known fact that a large scale general purpose quantum computer would be able to break virtually all public key protocols currently used on the internet. Given that the most optimistic estimates for the construction of the first practical and scalable quantum computer fall in the range of 10 to 15 years, one might wonder, why they are already causing such a burst in activity in the research community. To understand this, consider a talk by Brian Sniffen, a security engineer from Akamai Technology, at CRYPTO 2016[1]. Akamai is estimated to be responsible for 15-30% of all web traffic, which gives them the means to collect representative sample data from internet protocols in use. Sniffen argued, using examples from past breaks of protocols, that it will take at least 10 years to remove a broken cryptographic protocol from the public internet. The problem is that the internet is a large, organic, historically grown network and phasing out broken protocols is not an easy task in this environment. This means that post-quantum cryptographic protocols, which is what quantum resistant cryptography is often referred to as, need to be standardized and ready to ship at least 10 years before quantum computers that can break current cryptographic primitives are built

[1] https://www.youtube.com/watch?v=bAGiXimZ4kQ

in practice. As cryptographers we strive to err on the conservative side and thus we need to take into account the possibility that the optimistic estimates for practical quantum computers are accurate. If this is indeed the case then the time to roll out post-quantum protocols is now. Indeed, NIST has realized the need for such protocols and published a call for proposals[2]. Given that lattice-based cryptography is believed to be hard to break even for quantum computers, it is a natural candidate to replace current susceptible protocols.

Not surprisingly, there have been a huge number of proposals for lattice-based schemes to solve a number of cryptographic problems, for example [19, 3, 65, 44, 11, 20, 27, 10, 9, 26, 40, 12]. More and more of such schemes are moving from theoretical constructions to practical implementations [65, 47, 1, 67, 17, 31, 69, 46, 4]. This transition is peppered with a number of issues, most notably the one of concrete security. When choosing parameters for a cryptographic scheme, we need to ensure that the scheme is secure in a practical sense, i.e. that no realistic adversary can break it. Care must be taken to strike the right balance between security and performance, which are usually diametrical goals. In order to evaluate the amount of resources a hypothetical adversary would need to break a scheme, one usually analyzes how well the best known algorithms perform on typical instances. Since by definition one cannot solve secure instances with such algorithms, they are usually applied to small/easy instances and the results are extrapolated to larger/harder instances. As it turns out, lattice algorithms, i.e. algorithms solving geometric problems in lattices, are a useful tool in this context [2, 16, 45].

The conflict between efficiency and security is not specific to parameter selection, but also arises in other aspects of the implementation. Generally, lattice-based schemes are relatively easy to implement, since most of the operations consist of simple additions and multiplications of matrices and vectors over relatively small integers or ring elements.

[2] https://csrc.nist.gov/projects/post-quantum-cryptography

However, there is a caveat: many schemes assume randomness in a form that is not readily available on most platforms, which is crucial for the security proof. Converting an available source of randomness, usually uniform random bits, into the required distribution, commonly referred to as a sampling algorithm, can be tricky to implement efficiently and securely [64, 13, 68, 71]. Accordingly, most sampling algorithms do not produce the exact distribution, but rather approximate it. Naturally, the question arises, how close the approximation needs to be in order for the security proof to still hold. This needs to be addressed with rigor, as poor approximations have led to actual attacks [60]. The analysis needs to be carried out carefully, since otherwise it might lead to unsubstantiated security claims (see [70] for an example and [55] why it is incorrect).

Results

This thesis consists of multiple results tackling the issues outlined above. In Part I we explore recent progress on some popular lattice algorithms with a focus on their suitability for cryptanalysis. Part II is concerned with analyzing the security of implementations of sampling algorithms, both in general and for the specific case of lattice-based schemes.

Lattice Algorithms

The main topic of Chapter 3 is lattice point enumeration [36, 22], a classic algorithm to solve particular lattice problems relevant to cryptography. It can be used to solve several lattice problems, but in this thesis we focus on their ability to solve the Shortest Vector Problem (SVP) in a lattice. The problem can be stated as follows: given a matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$ with full column rank, find a vector $\mathbf{x} \in \mathbb{Z}^n \setminus \{0\}$ such that $\|\mathbf{Bx}\|$ is minimal. This problem is NP-hard [48] and the fastest known algorithms have exponential or even superexponential worst-case running time. Even though algorithms

with time and memory complexity of $2^{O(n)}$ exist, somewhat surprisingly, lattice point enumeration is the fastest known algorithm for practically relevant instances, despite having superexponential running time. This is often attributed to their small memory complexity. In the cryptanalytic context, they are commonly used to estimate lattice attacks on cryptographic schemes.

Enumeration consists of two steps: 1) preprocessing the input matrix $\mathbf{B}$ and 2) a brute-force search for the solution $\mathbf{x}$. (See Section 2.4.1 for a more detailed description.) It has been known for a long time that the size of the search space and thus the complexity of the second step, both asymptotically and in practice, depends heavily on the preprocessing step [33]. With regards to the asymptotic complexity, rigorous results were known, but they only pertained to the corner cases with very light [22] and extremely heavy preprocessing [36]. In particular, the algorithm of [22] uses a polynomial time algorithm to preprocess $\mathbf{B}$, which leads to $2^{O(n^2)}$ worst case complexity of the subsequent brute-force search. On the other hand, the algorithm of [36] uses a heavy recursive preprocessing, which is clearly also superexponential, but the search space for $\mathbf{x}$ can then be bounded by $n^{O(n)}$. Still, the latter step dominates the complexity of the entire algorithm asymptotically, so its running time is $n^{O(n)}$, which is clearly better than the aforementioned $2^{O(n^2)}$ bound for the algorithm of [22]. However, the hidden constants in the $n^{O(n)}$ bound seem to be too large to be useful in practice for achievable instances. So practical implementations usually employ some medium preprocessing, but its effect on the asymptotic behavior has been unclear for a long time. This is problematic for cryptanalysis, since one needs a sound asymptotic model in order to carry out the extrapolation as described above. Chapter 3 is, apart from a small extension, mostly concerned with analyzing the impact of medium preprocessing on the complexity of enumeration. We show that there are two different ways to obtain algorithms that are simultaneously competitive in practice and asymptotically efficient. They can each be viewed as generalizations of [22] and

[36], respectively. We discuss the impact of our results on cryptanalysis, in particular our generalization of [36]. For our generalization of [22], our results were applied in this context in a later survey [2].

In Chapter 4 we move on to a different type of algorithm: lattice block reduction. Block reduction algorithms are used in a similar context as enumeration. They get the same input, i.e. a matrix $\mathbf{B} \in \mathbb{R}^{m \times n}$, but they achieve different trade-offs with regards to running time and output quality. More specifically, while enumeration searches the entire search space for $\mathbf{x}$ and thus solves SVP exactly, block reduction algorithms approximate the shortest vector: they output a vector $\mathbf{x}' \in \mathbb{Z}^n$ such that $\|\mathbf{B}\mathbf{x}'\|$ is not too much larger than the minimal $\|\mathbf{B}\mathbf{x}\|$. In fact, how well they approximate the minimum is determined by a parameter $2 \leq k \leq n$, usually called the block size. Larger values of k lead to better approximation factors, which is the ratio $\|\mathbf{B}\mathbf{x}'\| / \|\mathbf{B}\mathbf{x}\|$, at the cost of longer running times. The trade-offs achievable with block reduction range from $\text{poly}(n)$ running time and $2^{O(n)}$ approximation factor for small k, to $n^{O(n)}$ running time (or $2^{O(n)}$ if exponential amounts of space are available) and constant approximation factor for large k close to the column rank of the input matrix n.

The two types of algorithms, block reduction and enumeration, are closely related: block reduction is often used as the preprocessing step in the enumeration, and they themselves often rely on enumeration in dimension k as a subroutine. In the context of cryptanalysis, block reduction is a popular tool and in many cases it is part of the best known attack to cryptographic schemes [2, 16]. Similarly as above, we need a solid theoretical understanding and efficient implementations of block reduction, in order to carry out meaningful extrapolation during the analysis. Unfortunately, up to recently, the reduction algorithms that were best understood in terms of asymptotic running time and output quality [23], were significantly outperformed by less understood algorithms [72]. Again, this is problematic for cryptanalysis. In Chapter 4 we present an algorithm

along with theoretical analysis and an experimental study that shows that it is possible to achieve both: a solid understanding of the asymptotics and approximation factor, and competitive running time. We rigorously analyze the worst-case output quality and running time of our algorithm. We then show that using a well established and common heuristic, namely the Gaussian Heuristic (see Section 2.4.3 for details), the analysis can be applied to the average case. This allows to predict its behavior simply using a closed formula. Through a large set of experiments we then show that our algorithm is competitive in practice and that the predictions are accurate.

We remark that many of the algorithms described in this part of the thesis are now part of the public lattice reduction library `fpLLL` [77].

Approximate Samplers

A common algorithmic task in many lattice-based schemes is to sample from a specific discrete distribution over the integers. There exists a range of algorithms with different time, memory, and randomness characteristics suitable for different settings [28, 14, 19, 37, 21, 63]. All of these algorithms, with the sole exception of [37], approximate the distribution, and their performance typically degrades with increasing precision, in one or more of their characteristics. Naturally, we want to set the precision just large enough to maintain the security of the scheme, in order to maximize its performance. To quantitatively analyze the precision-security trade-off, one usually reduces the security of the scheme that uses the approximate distribution to the security of the scheme that uses the ideal distribution, i.e. the distribution the sampler is trying to approximate. Such proofs are usually very generic and apply to entire classes of cryptographic schemes. A recent line of research [65, 5, 76] has demonstrated that in many settings there are surprisingly efficient reductions lowering the precision requirement for samplers and thus increasing their performance. A notable consequence is that in certain settings most of

the sampling algorithms may be implemented using 53-bit floating point numbers while maintaining meaningful security levels. Previously, this was only possible using much larger precision (say more than 100 bits), which is significantly slower on commodity computers. The results crucially rely on the notion of *bit security*, which is ubiquitous in the field of cryptography, but not formally defined. There seems to be a common understanding in the community of what bit security is meant to capture, a bound on the trade-off between resources and advantage of any adversary, but it has been noted that there are several problems with a straight-forward interpretation of this notion [7, 18].

In order to start building a sound theory around approximate samplers, we take a step back and propose a definition of bit security in Chapter 5. We believe the definition captures the security of a concrete instantiation of a scheme on an intuitive level. We leverage tools from information theory (while maintaining the computational context) to define an adversary's advantage as the amount of information it is able to extract. In line with the above intuition, this provides a lower bound on the amount of resources such an adversary requires to extract the entire secret. Surprisingly, our definition yields an expression that is either in line with, or diverges from, the common notion, depending on the flavor of the cryptographic primitive. We then justify our new definition by giving a series of tight reductions between schemes of different flavor, which, we believe, demonstrates that our definition is on target.

In Chapter 6 we apply our notion of bit security to approximate samplers, recovering previous results and extending them in several dimensions. Note that in order to analyze the trade-off between approximation quality and security guarantee, one needs to quantify the "distance" between the ideal distribution and its approximation. This is usually done using some divergence between probability distributions. The security guarantee then depends on 1) the bound on the divergence that can be obtained from common data types that are used for approximations, e.g. fixed point or floating point

numbers, and 2) how well the divergence lends itself to a security proof. In Chapter 6 we first analyze, which properties are required of a divergence to obtain an efficient security proof (along the lines of [65, 5, 76]). We then demonstrate how these properties can be used in a security proof, solidifying results from [65] (which contains gaps in the proof). Furthermore, the results of [65, 5, 76] only apply to a certain class of cryptographic primitives (one important example being signature schemes). Extending them to more general cryptographic primitives, e.g. arbitrary encryption schemes, has been an open problem so far, which we solve in this chapter using our definition of bit security. Finally, we present a new metric between probability distributions that is, to the best of our knowledge, unique in that it 1) is a metric (i.e. is symmetric and satisfies triangle inequality), 2) allows for efficient security proofs, and 3) is easily bounded by common data types, notably floating point numbers. We dub this new metric the *max-log* distance, for reasons that will become clear upon seeing its definition. It is aimed at simplifying security proofs for approximate samplers while maintaining their efficiency. We demonstrate its usefulness by applying it to our own sampling algorithm in the following chapter.

The final chapter of this thesis, Chapter 7, presents a new sampling algorithm. It has several advantages: 1) it is generic and thus can be used in any lattice-based scheme where such a sampler is required (note that many of the known samplers do not have this property), 2) it can be implemented very efficiently as we demonstrate in an experimental study, 3) we lay out how it can be easily implemented in constant time with minor or no performance penalty, a crucial feature in the context of cryptographic schemes and notoriously hard to achieve in this context [3, 68, 64].

Chapter 2

Background

Notation and Basic Linear Algebra

Throughout this thesis, the log refers to the logarithm with base 2 and ln to the one with base e. For $n \in \mathbb{Z}_+$ we denote the set $\{0,\dots,n\}$ by $[n]$. For vectors we use bold lower case letters and the i-th entry of a vector $\mathbf{v}$ is denoted by v_i. Occasionally, we construct vectors on the fly using the notation $(\cdot)_{i\in S}$ for some set S (or in short $(\cdot)_i$ if the set S is clear from context), where $\cdot$ is a function of i.

Matrices are denoted by bold upper case letters. The i-th column of a matrix $\mathbf{B}$ is denoted by $\mathbf{b}_i$. Furthermore, we denote the submatrix comprising the columns from the i-th to the j-th column (inclusive) as $\mathbf{B}_{[i,j]}$ and the horizontal concatenation of two matrices $\mathbf{B}_1$ and $\mathbf{B}_2$ by $[\mathbf{B}_1|\mathbf{B}_2]$.

Definition 1 *For two vectors* $\mathbf{v},\mathbf{w}$*, their* scalar product *is defined as*

$$\langle \mathbf{v},\mathbf{w}\rangle = \sum_i v_i \cdot w_i.$$

Definition 2 *For a vector* $\mathbf{v}$ *and some* $p \in \mathbb{R}_+$*, we define the* p norm *of* $\mathbf{v}$ *to be*

$$\|\mathbf{v}\|_p = \left(\sum |v_i|^p\right)^{1/p}.$$

Whenever we omit the subscript p, we mean the standard Euclidean norm*, i.e.* $p = 2$*. For any matrix* $\mathbf{B}$ *and* $p \geq 1$*, we define the* induced norm *to be* $\|\mathbf{B}\|_p = \max_{\|\mathbf{x}\|_p=1}(\|\mathbf{B}\mathbf{x}\|_p)$*. For* $p = 1$ *(resp.* ∞*) this is often denoted by the* column (row) sum norm*; for* $p = 2$ *this is also known as the* spectral norm*.*

Fact 1 *For any matrix* $\mathbf{B}$ *we have* $\|\mathbf{B}\|_2 \leq \sqrt{\|\mathbf{B}\|_1 \|\mathbf{B}\|_\infty}$.

Definition 3 *For two vectors* $\mathbf{b}$ *and* $\mathbf{v}$*, we define the* orthogonal projection *of* $\mathbf{v}$ *to* $\mathbf{b}$ *as*

$$\pi_{\mathbf{v}}(\mathbf{b}) = \mathbf{b} - \frac{\langle \mathbf{b}, \mathbf{v} \rangle}{\|\mathbf{v}\|^2}\mathbf{v}.$$

For two matrices $\mathbf{V}$ *and* $\mathbf{B}$*,* $\pi_{\mathbf{V}}(\mathbf{B})$ *is the matrix obtained by applying* $\pi_{\mathbf{V}}$ *to every column* $\mathbf{b}_i$ *of* $\mathbf{B}$*, where* $\pi_{\mathbf{V}}(\mathbf{b}_i) = \pi_{\mathbf{v}_k}(\cdots(\pi_{\mathbf{v}_1}(\mathbf{b}_i))\cdots)$.

The Gram-Schmidt process allows to compute a orthogonal basis of any linear space generated by a matrix $\mathbf{B}$.

Definition 4 *For each matrix* $\mathbf{B}$ *we define its* Gram-Schmidt-Orthogonalization *(GSO)* $\mathbf{B}^*$*, where the i-th column* $\mathbf{b}_i^*$ *of* $\mathbf{B}^*$ *is defined as* $\mathbf{b}_i^* = \pi_{\mathbf{B}^*_{[1,i-1]}}(\mathbf{b}_i) = \mathbf{b}_i - \sum_{j<i} \mu_{i,j}\mathbf{b}_j^*$ *and* $\mu_{i,j} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2$ *(and* $\mathbf{b}_1^* = \mathbf{b}_1$*).*

For a fixed matrix $\mathbf{B}$ we extend the projection operation to indices: $\pi_i(\cdot) = \pi_{\mathbf{B}^*_{[1,i-1]}}(\cdot)$, so $\pi_1(\mathbf{B}) = \mathbf{B}$. Whenever we refer to the *shape* of a matrix $\mathbf{B}$, we mean the vector $\mathbf{r} = (\|\mathbf{b}_i^*\|)_{i\in[n]}$. We define $\mathbf{B}^\dagger$ to be the GSO of $\mathbf{B}$ in reverse order, i.e. the matrix obtained by reversing the order of the columns of $\mathbf{B}$, applying the usual GSO, and reversing the order of the resulting matrix.

Arranging the values $\mu_{i,j}$ (cf. Definition 4) into an upper triangular matrix $\mathbf{M}$ in the obvious way and setting the diagonal elements to 1, we obtain a decomposition of $\mathbf{B} = \mathbf{B}^*\mathbf{M}$. Normalizing the columns of $\mathbf{B}^*$ we can further decompose it into $\mathbf{B}^* = \mathbf{Q}\mathbf{D}$,

where $\mathbf{D}$ is the diagonal matrix with the diagonal being the vector $\mathbf{r}$, and $\mathbf{Q}$ orthonormal. Now, $\mathbf{Q}$ and $\mathbf{R} = \mathbf{DM}$ comprise the usual QR decomposition of $\mathbf{B}$. When referring to the *GSO matrices*, we mean $\mathbf{D}$ and $\mathbf{M}$.

Probabilities and Information Theory

Calligraphic letters are reserved for probability distributions and $x \leftarrow \mathscr{P}$ means that x is sampled from the distribution $\mathscr{P}$. For any x in the support of $\mathscr{P}$ we denote its probability under $\mathscr{P}$ by $\mathscr{P}(x)$. All distributions in this work are discrete, and $\mathscr{U}(S)$ is the uniform distribution over the support S. If S is clear from context, we simply write $\mathscr{U}$ instead of $\mathscr{U}(S)$. The Bernoulli distribution with parameter p is denoted by $\mathscr{B}_p$.

Definition 5 *The* statistical distance *between two distributions $\mathscr{P}$ and $\mathscr{Q}$ over the same support S is defined as*

$$\Delta_{\mathrm{SD}}(\mathscr{P},\mathscr{Q}) = \frac{1}{2}\sum_{x\in S}|\mathscr{P}(x)-\mathscr{Q}(x)|.$$

Definition 6 *The* KL-divergence *between two distributions $\mathscr{P}$ and $\mathscr{Q}$ over the same support S is defined as*

$$\delta_{\mathrm{KL}}(\mathscr{P},\mathscr{Q}) = \sum_{x\in S}\mathscr{P}(x)\ln\frac{\mathscr{P}(x)}{\mathscr{Q}(x)}.$$

Note that the statistical distance is a metric, while the KL-divergence is not. Pinsker's inequality bounds Δ_{SD} in terms of δ_{KL}:

Fact 2 (Pinsker's inequality) *For any two distributions $\mathscr{P}$ and $\mathscr{Q}$ over the same support S we have $\Delta_{\mathrm{SD}}(\mathscr{P},\mathscr{Q}) \leq \sqrt{\delta_{\mathrm{KL}}(\mathscr{P},\mathscr{Q})/2}$.*

Definition 7 *A* probability ensemble *$\mathscr{P}_\theta$ is a family of distributions indexed by a parameter θ (which is possibly a vector). We extend any divergence δ between distributions to*

probability ensembles as $\delta(\mathscr{P}_\theta, \mathscr{Q}_\theta) = \max_\theta \delta(\mathscr{P}_\theta, \mathscr{Q}_\theta)$.

For notational simplicity, we do not make a distinction between random variables, probability distributions, and probabilistic algorithms generating them.

An algorithm A with oracle access to a sampler for distribution ensemble $\mathscr{P}_\theta$ is denoted by $A^{\mathscr{P}}$, which means that it adaptively sends queries θ_i to the sampler, which returns a sample from $\mathscr{P}_{\theta_i}$. If A uses only one sample from $\mathscr{P}_\theta$, then we write $A(\mathscr{P}_\theta)$.

We will need a few concepts from information theory when considering the security of cryptographic schemes in order to quantify the information an adversary is able to obtain about a secret (cf. Chapter 5).

Definition 8 *The* Shannon entropy *of a random variable X is given by*

$$H(X) = \mathbb{E}_X\left[\log \frac{1}{\Pr\{X\}}\right] = -\sum_x \Pr[X = x] \log \Pr[X = x].$$

Definition 9 *For two random variables X and Y, the* conditional entropy *of X given Y is*

$$H(X|Y) = \mathbb{E}_Y[H(X|Y)] = \sum_{x,y} \Pr[X = x, Y = y] \log \frac{\Pr[Y = y]}{\Pr[X = x, Y = y]}.$$

Definition 10 *The* mutual information *between two random variables X and Y is*

$$I(X;Y) = H(X) - H(X|Y).$$

Approximations

In this work we will occasionally encounter expressions of the form $\varepsilon + O(\varepsilon^2)$ for some small ε. In many of these cases, the constant c hidden in the asymptotic notation is much smaller than $1/\varepsilon$ (say $c\varepsilon \leq 2^{-30}$). So, the higher order term $O(\varepsilon^2)$

has virtually no impact, neither in practice nor asymptotically, on our applications. We define $\hat{\varepsilon} = \varepsilon + O(\varepsilon^2)$ and write $a \simeq b$ for $a = \hat{b}$, and similarly $a \lesssim b$ for $a \leq \hat{b}$. This allows us to drop the $O(\varepsilon^2)$ term in such cases and avoid tracing irrelevant terms through our calculations without losing rigor, e.g. $\ln(1+\varepsilon) = \varepsilon + O(\varepsilon^2)$ can be written as $\ln(1+\varepsilon) \simeq \varepsilon$.

A p-bit floating point (FP) approximation $\bar{x}$ of a real x stores the p most significant bits of x together with a binary exponent. This guarantees that the relative error, defined below, is bounded by $\leq 2^{-p}$.

Definition 11 *For a real x and an approximation $\bar{x} \in R$, we define the* relative error *as*

$$\delta_{\mathrm{RE}}(x, \bar{x}) = \frac{|x - \bar{x}|}{|x|}.$$

We extend the notion of relative error to any two distributions $\mathscr{P}$ and $\mathscr{Q}$ (and thus implicitly to probability ensembles)

$$\delta_{\mathrm{RE}}(\mathscr{P}, \mathscr{Q}) = \max_{x \in S} \delta_{\mathrm{RE}}(\mathscr{P}(x), \mathscr{Q}(x)) = \max_{x \in S} \frac{|\mathscr{P}(x) - \mathscr{Q}(x)|}{\mathscr{P}(x)},$$

where S is the support of $\mathscr{P}$.

The following fact is straightforward to verify:

Fact 3 *For any two distributions $\mathscr{P}$ and $\mathscr{Q}$ over the same support S we have*

$$\Delta_{\mathrm{SD}}(\mathscr{P}, \mathscr{Q}) \leq \frac{1}{2} \cdot \delta_{\mathrm{RE}}(\mathscr{P}, \mathscr{Q}).$$

The relative error can also be used to bound the KL-divergence:

Lemma 1 (Strengthening [65, Lemma 2]) *For any two distributions $\mathscr{P}$ and $\mathscr{Q}$ with the same support S with $\mu = \delta_{\mathrm{RE}}(\mathscr{P},\mathscr{Q}) < 1$,*

$$\delta_{\mathrm{KL}}(\mathscr{P},\mathscr{Q}) \leq \frac{\mu^2}{2(1-\mu)^2}.$$

In particular, if $\mu \leq 1/4$, then $\delta_{\mathrm{KL}}(\mathscr{P},\mathscr{Q}) \leq (8/9)\mu^2 < \mu^2$.

Proof Recall that $\delta_{\mathrm{KL}}(\mathscr{P},\mathscr{Q}) = \sum_i \mathscr{P}(i)\ln(\mathscr{P}(i)/\mathscr{Q}(i))$. For any $p,q > 0$, let $x = (p-q)/p = 1-(q/p) < 1$, so that $\ln(p/q) = -\ln(1-x) = x + e(x)$ with error function $e(x) = -x - \ln(1-x)$. Notice that $e(0) = 0$, $e'(0) = 0$ and $e''(x) = 1/(1-x)^2 \leq 1/(1-\mu)^2$ for all $x \leq \mu$. It follows that $e(x) \leq x^2/(2(1-\mu)^2) \leq \mu^2/(2(1-\mu)^2)$ for all $|x| \leq \mu$, and

$$\delta_{\mathrm{KL}}(\mathscr{P},\mathscr{Q}) = \sum_i \mathscr{P}(i)\ln\left(\frac{\mathscr{P}(i)}{\mathscr{Q}(i)}\right) \leq \sum_i \mathscr{P}(i)\cdot\left(\frac{\mathscr{P}(i)-\mathscr{Q}(i)}{\mathscr{P}(i)} + e\right) = 1 - 1 + e = e$$

where $e = \mu^2/(2(1-\mu)^2)$.□

This is a slight improvement over [65, Lemma 2], which shows that if $\mu \leq 1/4$, then $\delta_{\mathrm{KL}} \leq 2\mu^2$. So, Lemma 1 improves the bound by a constant factor $9/4$. In fact, for $\mu \approx 0$, Lemma 1 shows that the bound can be further improved to $\delta_{\mathrm{KL}} \lesssim \frac{1}{2}\mu^2$.

Lattices

We now define, and state some basic facts about, the main subject of this thesis: lattices.

Definition 12 *A* lattice Λ *is a discrete subgroup of* $\mathbb{R}^m$.

Fact 4 *Every lattice is generated by some matrix $\mathbf{B} \in \mathbb{R}^{m\times n}$, i.e. $\Lambda = \mathscr{L}(\mathbf{B}) = \{\mathbf{Bx} \colon \mathbf{x} \in$*

$\mathbb{Z}^n\}$. *If* $\mathbf{B}$ *has full column rank, it is called a* basis *of* Λ *and* $\dim(\Lambda) = n$ *is the* dimension *(or* rank*) of* Λ.

A lattice has infinitely many bases, which are related to each other by right-multiplication with unimodular matrices.

Definition 13 *A matrix* $\mathbf{U} \in \mathbb{Z}^{n \times n}$ *is called* unimodular*, if* $|\det(\mathbf{U})| = 1$.

Fact 5 *For any unimodular matrix* $\mathbf{U}$ *and basis* $\mathbf{B}$*, we have* $\mathscr{L}(\mathbf{B}) = \mathscr{L}(\mathbf{BU})$.

For every lattice Λ there are a few invariants associated to it.

Definition 14 *For any lattice* $\Lambda = \mathscr{L}(\mathbf{B})$*, its* determinant *is defined as* $\det(\mathscr{L}(\mathbf{B})) = \prod_i \|\mathbf{b}_i^*\|$.

The determinant is exactly the volume of the parallelepiped spanned by $\mathbf{B}$ and is also sometimes referred to as its *volume*. Even though the basis of a lattice is not uniquely defined, the determinant is and it is efficiently computable given a basis.

Definition 15 *For any lattice* Λ *we denote the length of its* shortest non-zero vector *(also known as the* first minimum*) by*

$$\lambda_1(\Lambda) = \min_{\{\mathbf{v} \in \Lambda \setminus \mathbf{0}\}} \|\mathbf{v}\|.$$

We define the problem of finding a shortest non-zero vector in a lattice given a basis of that lattice as the Shortest Vector Problem *(SVP).*

We use the short-hand notations $\det(\mathbf{B}) = \det(\mathscr{L}(\mathbf{B}))$ and $\lambda_1(\mathbf{B}) = \lambda_1(\mathscr{L}(\mathbf{B}))$. The two quantities are related by Hermite's constant.

Definition 16 *For every $n \in \mathbb{Z}_+$,* Hermite's constant *is defined to be*

$$\gamma_n = \max_{\{\Lambda | \dim(\Lambda)=n\}} \left(\frac{\lambda_1(\Lambda)}{\det(\Lambda)^{1/n}} \right)^2.$$

A classic result of Minkowski shows a bound on γ_n:

Fact 6 (Minkowski's Theorem) *For any $n \in \mathbb{Z}$, we have $\gamma_n \leq n$.*

More generally, one can show that $\gamma_n \in \Theta(n)$ [59].

Even though computing an upper bound on the length of a shortest vector is easy (as demonstrated by Minkowski's Theorem), solving SVP (even approximately) is NP-hard under randomized reductions [38, 49].

A useful concept in lattices (as in other fields of mathematics) is duality, which we introduce in the following.

Definition 17 *For every lattice Λ, its* dual *is defined as $\hat{\Lambda} = \{\mathbf{w} \in \text{span}(\Lambda) | \langle \mathbf{w}, \mathbf{v} \rangle \in \mathbb{Z}$ for all $\mathbf{v} \in \Lambda\}$.*

Fact 7 *For any lattice Λ we have $\det(\hat{\Lambda}) = \det(\Lambda)^{-1}$.*

Definition 18 *For a lattice basis $\mathbf{B}$ we define the* dual basis $\mathbf{D}$ *as the unique matrix that satisfies $\text{span}(\mathbf{B}) = \text{span}(\mathbf{D})$ and $\mathbf{B}^T\mathbf{D} = \mathbf{D}^T\mathbf{B} = \mathbf{I}$.*

The reason for this naming convention stems from the following fact:

Fact 8 *For a lattice basis $\mathbf{B}$ and its dual basis $\mathbf{D}$ we have $\widehat{\mathscr{L}(\mathbf{B})} = \mathscr{L}(\mathbf{D})$.*

Given a lattice basis, its dual basis is computable in polynomial time, but requires at least $\Omega(n^3)$ bit operations using matrix inversion. From Definition 18 follows that for any vector $\mathbf{w} = \mathbf{D}\mathbf{x}$ we have that $\mathbf{B}^T\mathbf{w} = \mathbf{x}$, i.e. we can recover the coefficients $\mathbf{x}$ of $\mathbf{w}$ with respect to the dual basis $\mathbf{D}$ by multiplication with the transpose of the primal basis $\mathbf{B}^T$. Finally, the following fact shows that the GSOs of a basis and its dual are related.

Fact 9 *For a lattice basis* $\mathbf{B}$ *and its dual* $\mathbf{D}$*, we have* $\|\mathbf{b}_i^*\| = 1/\|\mathbf{d}_i^\dagger\|$.

Enumeration

In order to solve SVP in practice, enumeration algorithms are usually employed, since these are the most efficient algorithms for currently realistic dimensions. The standard enumeration procedure, usually attributed to Fincke, Pohst [22], and Kannan [36] can be described as a recursive algorithm: given as input a basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$ and a radius r, it first recursively finds all vectors $\mathbf{v}' \in \mathscr{L}(\pi_2(\mathbf{B}))$ with $\|\mathbf{v}'\| \leq r$, and then for each of them finds all $\mathbf{v} \in \mathscr{L}(\mathbf{B})$, s.t. $\pi_2(\mathbf{v}) = \mathbf{v}'$ and $\|\mathbf{v}\| \leq r$, using $\mathbf{b}_1$. This essentially corresponds to a breadth first search on a large tree, where layers correspond to basis vectors and the nodes to the respective coefficients. While it is conceptually simpler to think of enumeration as a BFS, implementations usually employ a depth first search for performance reasons. Pseudo-code can be found in Algorithm 3.2 in Section 3.1.

At times during this thesis, a useful perspective on enumeration is to view it as iterating over the integer solutions of the linear system

$$\mathbf{B}\mathbf{x} = \mathbf{v} \tag{2.1}$$

such that $\|\mathbf{v}\| \leq r$. For this, observe that if $\mathbf{B} = \mathbf{Q}\mathbf{R}$ is the QR decomposition of $\mathbf{B}$, we have $\|\mathbf{B}\mathbf{x}\| = \|\mathbf{Q}\mathbf{R}\mathbf{x}\| = \|\mathbf{R}\mathbf{x}\|$. Accordingly, we focus on the system $\mathbf{R}\mathbf{x} = \mathbf{w}$ such that $\|\mathbf{w}\| \leq r$. Now recall from the recursive description that the problem is first solved in the projected lattice $\pi_2(\mathbf{R})$. Since $\mathbf{R}$ is upper triangular, this simply corresponds to dropping the first column and row. Once we have collected all valid solutions (meaning that the projected vectors are shorter than r) recursively for the reduced system, we can easily generate for each of them a list of solutions for the original system using the first row of $\mathbf{R}$. Unrolling the recursion, we see that enumeration starts with finding all integer

solutions for the last coordinate of $\mathbf{x}$, which are passed to the recursion level above, etc. The triangular structure of $\mathbf{R}$ makes this procedure reminiscent of an "iterated backward substitution" from the usual Gaussian elimination.

The complexity, both in an asymptotic and in a concrete sense, of enumeration depends heavily on the shape of the input basis [36, 35, 33]. Accordingly, enumeration is usually combined with a preprocessing step to obtain an asymptotic complexity bound and speed up the algorithm in practice. Unfortunately, the types of preprocessing that yield the best overall asymptotic complexity (currently $O(n^{n/2e})$ [33]) are rarely used in practice as the hidden constants are too large to yield competitive running times in practice for currently tractable dimensions. We will explore this gap between theory and practice more in Chapter 3 and develop algorithms that are both, asymptotically and practically efficient.

There are several practical improvements of enumeration collectively known as *SchnorrEuchner enumeration* [74]: First, due to the symmetry of lattices, we can reduce the search space by ensuring that the last non-zero coefficient is always positive. Furthermore, if we find a vector shorter than the bound r, we can update the latter. And finally, we can enumerate the coefficients of a basis vector in order of the length of the resulting (projected) vector and thus increase the chance of finding some short vector early, which will update the bound r and keep the search space smaller.

It has also been demonstrated [25] that reducing the search space (and thus the success probability) – a technique known as pruning – can speed up enumeration by exponential (but lower order) factors. For more details on recent improvements we refer to [25, 35, 33, 52, 79].

Lattice Reduction

We classify lattice reduction into two categories: global reductions and block reductions. Global reductions yield in some sense the best possible basis. Block reductions apply these global reductions to smaller blocks of the basis in some sequence, in order to improve the overall quality of the basis.

Global Reductions

For every lattice basis $\mathbf{B}$ there are infinitely many bases that have the same shape, among which there is a (not necessarily unique) basis that minimizes $\|\mathbf{b}_i\|$ for all i. This is equivalent to the condition that the GSO coefficients satisfy $|\mu_{i,j}| \leq 1/2$ for all $i > j$. Transforming a basis into this form is commonly known as *size reduction* and is easily and efficiently done using a slight modification of the Gram-Schmidt process. In this work we will implicitly assume all bases to be size reduced. The reader can simply assume that any basis operation described in this work is followed by a size reduction.

We will often modify a lattice basis $\mathbf{B}$ such that its first vector satisfies $\alpha\|\mathbf{b}_1\| \leq \lambda_1(\mathbf{B})$ for some $\alpha \leq 1$. We will call this process *SVP reduction* of $\mathbf{B}$. Given an SVP oracle, it can be accomplished by using the oracle to find the shortest vector in $\mathscr{L}(\mathbf{B})$, prepending it to the basis, and running LLL (cf. Section 2.4.2) on the resulting generating system to remove the linear dependencies. In the context of reduction algorithms, the relaxation factor α is usually needed for proofs of termination or running time and only impacts the analysis of the output quality in lower order terms. In this work, we will sweep it under the rug and take it implicitly to be a constant close to 1. Finally, we will apply SVP reduction to projected blocks of a basis $\mathbf{B}$, for example we will SVP reduce the block $\pi_i(\mathbf{B}_{[i,i+k]})$. By that we mean that we will modify $\mathbf{B}$ in such a way that $\pi_i(\mathbf{B}_{[i,i+k]})$ is SVP reduced. This can easily be achieved by applying the transformations to the original basis vectors instead of their projections. If $\mathbf{B}$ is such that $\pi_i(\mathbf{B})$ is SVP

reduced for all i, then it is called *HKZ* reduced.

Analogously to the primal case above, we can modify a basis $\mathbf{B}$ such that its dual $\mathbf{D}$ satisfies $\alpha\|\mathbf{d}_n\| \leq \lambda_1(\widehat{\mathscr{L}(\mathbf{B})})$, i.e. its reversed dual basis is SVP reduced. This process is called *dual SVP reduction*. Again, for simplicity we will take α implicitly to be a constant close to 1. Note that if $\mathbf{B}$ is dual SVP reduced, then $\|\mathbf{b}_n^*\|$ is maximal among all bases of $\mathscr{L}(\mathbf{B})$. The obvious way to achieve dual SVP reduction is to compute the dual of the basis, SVP reduce it as described above, and compute the primal basis. While the transition between primal and dual basis is a polynomial time computation, it involves matrix inversion, which can be quite time consuming in practice. To address this issue, Gama and Nguyen [23] proposed a different strategy. SVP reduction, as performed by enumeration, consists of two steps: 1) the coordinates of a shortest vector in the given basis are computed, and 2) this vector is inserted into the basis. Note that the enumeration procedure (step 1) only operates on the GSO matrices of the basis (cf. Section 2.4.1) so it is sufficient for step 1 to invert the GSO matrices of the projected block, which is considerably easier since they consist of a diagonal and an upper triangular matrix. Furthermore, Gama and Nguyen observe that for dual SVP reduction, step 2 can be accomplished using the coordinates obtained during the dual enumeration by solely operating on the primal basis. Overall, their strategy is much more efficient in practice, but we remark that step 1 still incurs a computational overhead of $\Omega(n^3)$. We present an alternative way to carry out step 1 in Section 3.1, which will not require to compute any additional information about the dual basis. We achieve this by designing a "dual enumeration" algorithm, which strongly resembles the traditional enumeration procedure and is just as efficient.

Block Reduction

As opposed to global reduction algorithms, block reductions do not find the optimal basis but rather approximate it. The quality of their output is usually measured in the length of the shortest vector they are able to find with respect to lattice invariants: the *approximation factor* $\alpha = \|\mathbf{b}_1\|/\lambda_1(\mathbf{B})$ quantifies the length of the first vector in terms of the first minimum, while the *Hermite factor* $\bar{\delta} = \|\mathbf{b}_1\|/\det(\mathbf{B})^{1/n}$ expresses it in terms of the root determinant, i.e. the normalized density, of the lattice. The Hermite factor depends on the lattice dimension n, but the experiments of [24] suggest that in practice the *root Hermite factor* $\delta = \bar{\delta}^{1/n}$ converges to a constant as n increases for popular reduction algorithms. During our experiments we found that to be true at least for large enough dimensions ($n \geq 140$) [54].

The *LLL* algorithm [41] is a polynomial time basis reduction algorithm. A basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$ can be defined to be LLL reduced if $\mathbf{B}_{[1,2]}$ is SVP reduced and $\pi_2(\mathbf{B})$ is LLL reduced. From this it is straight forward to prove that LLL reduction achieves a root Hermite factor of at most $\delta \leq \gamma_2^{1/4} \approx 1.0746$. However, LLL has been reported to behave much better in practice [61, 24].

Using LLL reduction, we can define a slight relaxation of HKZ reduction [33, 35, 36]: we call a basis $\mathbf{B}$ *quasi-HKZ* reduced, if it is LLL reduced and $\pi_1(\mathbf{B})$ is HKZ reduced.

BKZ [72, 74] is a generalization of LLL to larger block size. A basis $\mathbf{B}$ is BKZ reduced with block size k (denoted by BKZ-k) if $\mathbf{B}_{[1,\min(k,n)]}$ is SVP reduced and $\pi_2(\mathbf{B})$ is BKZ-k reduced. BKZ achieves this by simply scanning the basis from left to right and SVP reducing each projected block of size k (or smaller once it reaches the end) by utilizing a SVP oracle for all dimensions $\leq k$. It iterates this process (which is usually called a *tour*) until no more change occurs. The following bounds for $\mathbf{b}_1$ of a BKZ-k

reduced basis hold [32]:

$$\|\mathbf{b}_1\| \leq \gamma_k^{\frac{n-1}{k-1}} \lambda_1(\mathbf{B}) \qquad \leq k^{\frac{n-1}{k-1}} \lambda_1(\mathbf{B}) \tag{2.2}$$

$$\|\mathbf{b}_1\| \leq 2\gamma_k^{\frac{n-1}{2(k-1)}+\frac{3}{2}} \det(\mathbf{B})^{1/n} \leq k^{\frac{n-1}{2(k-1)}+\frac{3}{2}} \det(\mathbf{B})^{1/n} \tag{2.3}$$

where the latter inequalities follow from known bounds on γ_k, respectively [59]. Inequality (2.3) shows that the root Hermite factor achieved by BKZ-k is at most $\lesssim \gamma_k^{\frac{1}{2(k-1)}}$. Furthermore, while there is no polynomial bound on the number of calls BKZ makes to the SVP oracle, Hanrot, Pujol, and Stehlé showed in [32] that one can terminate BKZ after a polynomial number of calls to the SVP oracle and still provably achieve the bound (2.3). Finally, BKZ has been repeatedly reported to behave very well in practice [24, 16]. For these reasons, BKZ is very popular in practice and implementations are readily available in different libraries, e.g. in `NTL`[75] or `fpLLL`[77].

In [23], Gama and Nguyen introduced a different block reduction algorithm, namely *Slide reduction*. It is also parameterized by a block size k, which is required to divide the lattice dimension n, but uses a SVP oracle only in dimension k.[1] A basis $\mathbf{B}$ is defined to be slide reduced with block size k, if $\mathbf{B}_{[1,k]}$ is SVP reduced, $\pi_2(\mathbf{B}_{[2,k+1]})$ is dual SVP reduced (if $k < n$), and $\pi_{k+1}(\mathbf{B}_{[k+1,n]})$ is slide reduced. Slide reduction, as described in [23], reduces a basis by first alternately SVP reducing all blocks $\pi_{ik+1}(\mathbf{B}_{[ik+1,(i+1)k]})$ and running LLL on $\mathbf{B}$. Once no more changes occur, the blocks $\pi_{ik+2}(\mathbf{B}_{[ik+2,(i+1)k+1]})$ are dual SVP reduced. This entire process is iterated until no more changes occur. Upon

[1] Strictly speaking, the algorithm as described in [23] uses HKZ reduction and thus requires an SVP oracle in lower dimensions as well. However, the entire analysis in [23] only relies on the SVP reducedness of the projected blocks and thus the HKZ reduction can be replaced by SVP reduction, which we do in the following.

termination, the basis is guaranteed to satisfy

$$\|\mathbf{b}_1\| \leq \gamma_k^{\frac{n-k}{k-1}} \lambda_1(\mathbf{B}) \qquad \leq k^{\frac{n-k}{k-1}} \lambda_1(\mathbf{B}) \tag{2.4}$$

$$\|\mathbf{b}_1\| \leq \gamma_k^{\frac{n-1}{2(k-1)}} \det(\mathbf{B})^{1/n} \leq k^{\frac{n-1}{2(k-1)}} \det(\mathbf{B})^{1/n} \tag{2.5}$$

This is slightly better than inequality (2.2) and (2.3), but the achieved root Hermite factor is also only guaranteed to be less than $\gamma_k^{\frac{1}{2(k-1)}}$. Slide reduction has the desirable properties of only making a polynomial number of calls to the SVP oracle and that all calls are in dimension k (and not in lower dimensions). The latter allows for a cleaner analysis, for example when combined with the Gaussian Heuristic (cf. Section 2.4.3). Unfortunately, Slide reduction has been reported to be greatly inferior to BKZ in experiments [24], so it is rarely used in practice and we are not aware of any publicly available implementation. This only changed in the course of this work, when we contributed our implementation to `fpLLL`, which resulted, among others, in the first public implementation of Slide reduction.

The Gaussian Heuristic

The Gaussian Heuristic gives an approximation of the number of lattice points in a "nice" subset of $\mathbb{R}^n$. More specifically, it says that for a given set S and a lattice Λ, we have $|S \cap \Lambda| \approx \mathrm{vol}(S)/\det(\Lambda)$. The heuristic has proved to be very useful in the average case analysis of lattice algorithms. For example, it can be used to estimate the complexity of enumeration algorithms [25, 33] or the output quality of lattice reduction algorithms [16]. For the latter, note that reduction algorithms work by repeatedly computing the shortest vector in some lattice and inserting this vector in a certain position of the basis. To estimate the effect such a step has on the basis, it is useful to be able to predict how long such a vector might be. This is where the Gaussian Heuristic comes in: using the

above formula, one can estimate how large the radius of an n-dimensional ball (this is the "nice" set) needs to be such that we can expect it to contain a non-zero lattice point (where $n = \dim(\Lambda)$). Using the volume formula for the n-dimensional ball, we get an estimate for the shortest non-zero vector in a lattice Λ:

$$GH(\Lambda) = \frac{(\Gamma(n/2+1) \cdot \det(\Lambda))^{1/n}}{\sqrt{\pi}} \tag{2.6}$$

If k is an integer, we define $GH(k)$ to be the Gaussian Heuristic (i.e. equation (2.6)) for k-dimensional lattices with unit determinant. The heuristic has been tested experimentally [25], also in the context of lattice reduction [24, 16], and been found to be too rough in small dimensions, but to be quite accurate starting in dimension > 45. In fact, for a precise definition of random lattices (which we are not concerned with in this work) it can be shown that the expected value of the first minimum of the lattice (over the choice of the lattice) converges to equation (2.6) as the lattice dimension tends to infinity.[2]

Heuristic 1 [Gaussian Heuristic] *For a given lattice* Λ, $\lambda_1(\Lambda) = GH(\Lambda)$.

Invoking Heuristic 1 for all projected sublattices that the SVP oracle is called on during the process, the root Hermite factor achieved by lattice reduction (usually with regards to BKZ) is commonly estimated to be [2]

$$\delta \approx GH(k)^{\frac{1}{k-1}}. \tag{2.7}$$

However, since the Gaussian Heuristic only seems to hold in large enough dimensions and BKZ makes calls to SVP oracles in all dimensions up to the block size k, it is not

[2] One can also formulate Heuristic 1 for a given lattice by assuming it "behaves like a random lattice". Depending on the exact definition of what it means for a lattice to "behave like a random lattice", this version is either stronger as or equivalent to Heuristic 1.

immediately clear how justified this estimation is. While there is a proof by Chen [15] that under the Gaussian Heuristic, equation (2.7) is accurate for BKZ, this is only true as the lattice dimension tends to infinity. It might be reasonable to assume that this also holds in practice as long as the lattice dimension is large enough compared to the block size, but in cryptanalytic settings this is often not the case. In fact, in order to achieve an approximation good enough to break a cryptosystem, a block size at least linear in the lattice dimension is often required. As another approach to predicting the output of BKZ, Chen and Nguyen proposed a simulation routine [16]. We provide a public implementation [78]. Unfortunately, the simulator approach has several drawbacks. Obviously, it requires more effort to apply than a closed formula like (2.7), since it needs to be implemented and "typical" inputs need to be generated or synthesized (among others, the shape of a "typical" HKZ reduced basis in dimension 45). On top of that, the accuracy of the simulator is based on several additional heuristic assumptions, the validity of which has not been independently verified.

To the best of our knowledge there have been no attempts to make similar predictions for Slide reduction, as it is believed to be inferior to BKZ and thus usually not considered for cryptanalysis.

Discrete Gaussians

Discrete Gaussians, a discretized version of the classic Gaussian distribution, play an important role in lattice based cryptography. In this work, we will only be concerned with one dimensional Gaussians. For the general case we refer the reader to [51].

Definition 19 *The* Gaussian function $\rho : \mathbb{R} \mapsto \mathbb{R}_+$ *is defined as* $\rho(x) = \exp(-\pi x^2)$. *We extend it to countable sets* $S \subset \mathbb{R}$ *by* $\rho(S) = \sum_{x \in S} \rho(x)$. *We write* $\rho_{c,s}(x) = \rho((x-c)/s)$ *for the Gaussian function centered around* $c \in \mathbb{R}$ *and scaled by a factor* $s \in \mathbb{R}$.

Definition 20 *For a countable set $S \subset \mathbb{R}$, $c, s \in \mathbb{R}$, the* discrete Gaussian distribution *$\mathscr{D}_{S,c,s}$ is the distribution that samples $y \leftarrow \mathscr{D}_{S,c,s}$ with probability*

$$\mathscr{D}_{S,c,s}(y) = \rho_{c,s}(y)/\rho_{c,s}(S)$$

for any $y \in S$.

Throughout this work, the subset S will usually be the integers or a coset thereof. Sampling from $\mathscr{D}_{\mathbb{Z},c,s}$ is computationally equivalent to sampling from $\mathscr{D}_{c+\mathbb{Z},s}$, the centered discrete Gaussian over the coset $c+\mathbb{Z}$. It has been shown that above a certain threshold for s, the discrete Gaussian behaves similarly to the continuous one. To quantify this behavior, [51] introduced the smoothing parameter.

Definition 21 *For any $\varepsilon > 0$, the* smoothing parameter *of the integers $\eta_\varepsilon(\mathbb{Z})$ is the smallest $s > 0$ such that $\rho(s\mathbb{Z}) \leq 1+\varepsilon$.*

A special case of [51, Lemma 3.3] shows an upper bound on the smoothing parameter of the integers.

Fact 10 *For any $\varepsilon > 0$,*

$$\eta_\varepsilon(\mathbb{Z}) \leq \sqrt{\ln(2+2/\varepsilon)/\pi}.$$

So, $\eta_\varepsilon(\mathbb{Z}) < 6$ is a relatively small constant even for very small values of $\varepsilon < 2^{-160}$. Another useful bound, which easily follows from Poisson summation formula [51, Lemma 2.8], is $\delta_{\mathrm{RE}}(s, \rho_{c,s}(\mathbb{Z})) \leq \delta_{\mathrm{RE}}(s, \rho_s(\mathbb{Z})) = \rho(s\mathbb{Z}) - 1$. This implies the following fact.

Fact 11 *For any $s \geq \eta_\varepsilon(\mathbb{Z})$, and $c \in \mathbb{R}$, we have*

$$\delta_{\mathrm{RE}}(s, \rho_{c,s}(\mathbb{Z})) \leq \varepsilon,$$

i.e., the total measure of $\rho_{c,s}(\mathbb{Z})$ approximates s.

We will use the smoothing parameter to invoke the following tail bound.

Lemma 2 ([28, Lemma 4.2 (ePrint)]) *For any $\varepsilon > 0$, any $s > \eta_\varepsilon(\mathbb{Z})$, and any $t > 0$,*

$$\Pr_{x \leftarrow \mathscr{D}_{\mathbb{Z},c,s}}[|x-c| \geq t \cdot s] \leq 2e^{-\pi t^2} \cdot \frac{1+\varepsilon}{1-\varepsilon}.$$

Finally, we will also require the smoothing parameter for the following discrete convolution theorems, specialized to the one dimensional case.

Theorem 1 ([50, Theorem 3]) *Let $\mathbf{z} \in \mathbb{Z}^m$ a nonzero integer vector, $\mathbf{s} \in \mathbb{R}^m$ with $s_i \geq \sqrt{2}\|\mathbf{z}\|_\infty \eta_\varepsilon(\mathbb{Z})$ for all $i \leq m$ and $c_i + \mathbb{Z}$ arbitrary cosets of $\mathbb{Z}$. Let y_i be independent samples from $\mathscr{D}_{c_i+\mathbb{Z},s_i}$, respectively. Then the distribution of $y = \sum z_i y_i$ is close to $\mathscr{D}_{Y,s}$, where $Y = \sum_i z_i c_i + \gcd(\mathbf{z})\mathbb{Z}$ and $s = \sqrt{\sum_i z_i^2 s_i^2}$. In particular, if $\tilde{\mathscr{D}}_{Y,s}$ is the marginal distribution of y, then $\delta_{\mathrm{RE}}(\mathscr{D}_{Y,s}, \tilde{\mathscr{D}}_{Y,s}) \leq \frac{1+\varepsilon}{1-\varepsilon} - 1 \simeq 2\varepsilon$.*

Theorem 2 ([63, Theorem 1]) *Let $s_1, s_2 > 0$, with $s^2 = s_1^2 + s_2^2$ and $s_3^{-2} = s_1^{-2} + s_2^{-2}$. Let $\Lambda = K\mathbb{Z}$ be a copy of the integer lattice $\mathbb{Z}$ scaled by a constant K. For any c_1 and $c_2 \in \mathbb{R}$, denote the distribution of $x_1 \leftarrow x_2 + \mathscr{D}_{c_1-x_2+\mathbb{Z},s_1}$, where $x_2 \leftarrow \mathscr{D}_{c_2+\Lambda,s_2}$, by $\tilde{\mathscr{D}}_{c_1+\mathbb{Z},s}$. If $s_1 \geq \eta_\varepsilon(\mathbb{Z})$, $s_3 \geq \eta_\varepsilon(\Lambda) = K\eta_\varepsilon(\mathbb{Z})$, then*

$$\delta_{\mathrm{RE}}(\mathscr{D}_{c_1+\mathbb{Z},s}, \tilde{\mathscr{D}}_{c_1+\mathbb{Z},s}) \leq \left(\frac{1+\varepsilon}{1-\varepsilon}\right)^2 - 1 \simeq 4\varepsilon$$

.

Part I

Lattice Algorithms

Chapter 3

Enumeration

This chapter explores several extensions and improvements of the traditional enumeration algorithm. We will first introduce a way to enumerate all short vectors in the dual of a lattice (specified by a given basis) without the need to compute neither the dual basis nor its GSO. In the following sections we will analyze the impact of preprocessing the basis on the complexity of the enumeration step. This will allow us to obtain variants that are both, asymptotically and practically efficient, and hint at the impact of these variants on cryptanalysis.

Dual Enumeration

Let $\mathbf{B}$ be a lattice basis and $\mathbf{D}$ the corresponding dual basis. The goal in this section is to efficiently find all vectors $\mathbf{x}$ such that $\|\mathbf{Dx}\| \leq r$ for some given $r \geq \lambda_1(\mathbf{D})$ by enumeration while only accessing $\mathbf{B}$.

Recall from Section 2.4.1 that we can view enumeration on the dual basis $\mathbf{D}$ as iterating over the integer solutions of the linear system

$$\mathbf{Dx} = \mathbf{v} \tag{3.1}$$

which can be simplified to $\mathbf{R_D x} = \mathbf{v}$, where $\mathbf{D} = \mathbf{Q_D R_D}$ is the QR decomposition of $\mathbf{D}$.

Unfortunately, computing $\mathbf{R_D}$ from $\mathbf{B}$ requires $O(n^3)$ operations, but multiplying (3.1) on the left by $\mathbf{B}^T$ yields the system

$$\mathbf{x} = \mathbf{B}^T \mathbf{v}. \tag{3.2}$$

Again considering the QR decomposition, this time of $\mathbf{B} = \mathbf{Q_B}\mathbf{R_B}$, we obtain $\mathbf{x} = \mathbf{R_B^T}\mathbf{Q_B^T}\mathbf{v}$. Since again $\|\mathbf{Q_B^T}\mathbf{v}\| = \|\mathbf{v}\|$, we can focus on the system $\mathbf{x} = \mathbf{R_B^T}\mathbf{w}$ and find all integer solutions such that $\|\mathbf{w}\| \leq r$. Because $\mathbf{R_B^T}$ is lower triangular, the solutions to the system can again be efficiently generated by an "iterated substitution", which now starts with the first entry of $\mathbf{x}$. The following lemma makes this intuition more explicit and will allow us to derive a concrete enumeration procedure for the dual of a lattice.

Lemma 3 *Let* $\mathbf{B}$ *be a lattice basis and* $\mathbf{w}$ *an arbitrary vector in the linear span of* $\mathbf{B}$*. Let* $\mathbf{x}$ *be the coefficient vector expressing* $\mathbf{w}$ *with respect to the dual basis, i.e.,* $x_i = \langle \mathbf{w}, \mathbf{b}_i \rangle$ *for all* $i \leq n$*. Then, for any* $k \leq n$*, the (uniquely defined) vector* $\mathbf{w}^{(k)} \in \mathrm{span}(\mathbf{B}_{[1,k]})$ *such that* $\langle \mathbf{w}^{(k)}, \mathbf{b}_i \rangle = x_i$ *for all* $i \leq k$*, can be expressed as* $\mathbf{w}^{(k)} = \sum_{i \leq k} \alpha_i \mathbf{b}_i^* / \|\mathbf{b}_i^*\|^2$ *where*

$$\alpha_i = x_i - \sum_{j<i} \mu_{i,j} \alpha_j. \tag{3.3}$$

Proof The condition $\mathbf{w}^{(k)} \in \mathrm{span}(\mathbf{B}_{[1,k]})$ directly follows from the definition of $\mathbf{w}^{(k)} = \sum_{i \leq k} \alpha_i \mathbf{b}_i^* / \|\mathbf{b}_i^*\|^2$. We need to show that this vector also satisfies the scalar product conditions $\langle \mathbf{w}^{(k)}, \mathbf{b}_i \rangle = x_i$ for all $i \leq k$. Substituting the expression for $\mathbf{w}^{(k)}$ in the scalar product we get

$$\langle \mathbf{w}^{(k)}, \mathbf{b}_i \rangle = \sum_{j \leq k} \alpha_j \frac{\langle \mathbf{b}_j^*, \mathbf{b}_i \rangle}{\|\mathbf{b}_j^*\|^2} = \sum_{j \leq i} \alpha_j \frac{\langle \mathbf{b}_j^*, \mathbf{b}_i \rangle}{\|\mathbf{b}_j^*\|^2} = \alpha_i + \sum_{j<i} \alpha_j \mu_{i,j} = x_i$$

where the last equality follows from the definition of α_i. □

This shows that if we enumerate the levels from $k = 1$ to n (note the reverse order as opposed to primal enumeration) we can easily compute α_k from all the given or previously computed quantities in $O(n)$. The length of $\mathbf{w}^{(k)}$ is given by

$$\|\mathbf{w}^{(k)}\|^2 = \sum_{i \leq k} \alpha_i^2 / \|\mathbf{b}_i^*\|^2 = \|\mathbf{w}^{(k-1)}\|^2 + \alpha_k^2 / \|\mathbf{b}_k^*\|^2. \tag{3.4}$$

To obtain an algorithm that is practically as efficient as primal enumeration, it is necessary to apply the same standard optimizations of SchnorrEuchner enumeration to the dual enumeration. It is obvious that we can exploit lattice symmetry and dynamic radius updates in the same fashion as in the primal enumeration. The only optimization that is not entirely obvious is enumerating the values for x_k in order of increasing length of the resulting partial solution. However, from Equation (3.3) and (3.4) it is clear that we can start by selecting $x_k = \lfloor \sum_{j<k} \mu_{k,j} \alpha_j \rceil$ in order to minimize the first value of α_k, and then proceed by alternating around this first value just as in the SchnorrEuchner primal enumeration algorithm.

It is also noteworthy that being able to compute partial solutions even allows us to apply pruning [25] directly. In summary this shows that dual SVP enumeration should be just as efficient as primal enumeration. To illustrate this, Algorithm 3.1 and 3.2 show the SchnorrEuchner variant of the two enumeration procedures.[1]

[1] The function `nextX` simply selects the next value for a specific variable in order to alternate correctly around the center of the interval of valid values. We omit details here since it works identical in both algorithms and requires auxiliary variables that would clutter the code unnecessarily.

Algorithm 3.1. Dual Enumeration

`procedure` DualEnum(μ, $(\|\mathbf{b}_i^*\|^2)_i$, A)

Input: The GSO of a lattice μ and $(\|\mathbf{b}_i^*\|^2)_{i\in[n]}$ and an upper bound A to the squared length of a shortest dual vector

Output: The coordinates of a shortest dual vector in the dual basis $\mathbf{D}$

1 $k \leftarrow 1$
2 `while` $k \geq 1$
3 $\quad\alpha_k \leftarrow x_k - \sum_{j<k} \mu_{k,j}\alpha_j$
4 $\quad l_k \leftarrow l_{k-1} + \alpha_k^2/\|\mathbf{b}_k^*\|^2$
5 $\quad$`if` $l_k \leq A$ and $k = n$ `then`
6 $\quad\quad\mathbf{s} \leftarrow \mathbf{x}, A \leftarrow l_k$
7 $\quad$`if` $l_k \leq A$ and $k < n$ `then`
8 $\quad\quad k \leftarrow k+1, x_k \leftarrow \lfloor \sum_{j<k} \mu_{k,j}\alpha_j \rceil$
9 $\quad$`else`
10 $\quad\quad k \leftarrow k-1, x_k \leftarrow \text{nextX}(k)$
11 `return` $\mathbf{s}$

Algorithm 3.2. Primal Enumeration

`procedure` PrimalEnum(μ, $(\|\mathbf{b}_i^*\|^2)_i$, A)

Input: The GSO of a lattice μ and $(\|\mathbf{b}_i^*\|^2)_{i\in[n]}$ and an upper bound A to the squared length of a shortest vector

Output: The coordinates of a shortest vector in the basis $\mathbf{B}$

1 $k \leftarrow n$
2 `while` $k \leq n$
3 $\quad\alpha_k \leftarrow x_k + \sum_{j>k} \mu_{j,k}x_j$
4 $\quad l_k \leftarrow l_{k+1} + \alpha_k^2\|\mathbf{b}_k^*\|^2$
5 $\quad$`if` $l_k \leq A$ and $k = 1$ `then`
6 $\quad\quad\mathbf{s} \leftarrow \mathbf{x}, A \leftarrow l_k$
7 $\quad$`if` $l_k \leq A$ and $k > 1$ `then`
8 $\quad\quad k \leftarrow k-1, x_k \leftarrow \lfloor -\sum_{j>k} \mu_{j,k}x_j \rceil$
9 $\quad$`else`
10 $\quad\quad k \leftarrow k+1, x_k \leftarrow \text{nextX}(k)$
11 `return` $\mathbf{s}$

The Complexity of Enumeration

We now turn to the impact of preprocessing in the enumeration. For this we first introduce a new notion of basis reduction that will facilitate the analysis later.

Definition 22 *Let* $\mathbf{B} \in \mathbb{Z}^{m\times n}$, $n' = \dim(\mathscr{L}(\mathbf{B}))$, *and* $\zeta : [n] \to \mathbb{R}_+$. *We call* $\mathbf{B}$ ζ-reduced,

if for all $i \in [n]$

$$\|\mathbf{b}_i^*\| > \zeta(i)\det(\mathbf{B})^{1/n'} \quad \Rightarrow \quad \lambda_1(\pi_{i-1}(\mathbf{B})) > \lambda_1(\mathbf{B})$$

and $\mathbf{B}_{[1,k]}$ *is* ζ*-reduced for all* $k \in [n-1]$.

Note that the definition covers arbitrary generating systems for lattices, not only bases. The quantification over the sublattices in Definition 22 might seem like a stringent condition at first sight but the reduction algorithms that we consider, namely LLL, BKZ and HKZ reduction, naturally meet this condition since all subbases of the form $\mathbf{B}_{[1,k]}$ of a reduced bases $\mathbf{B}$ are also reduced. Next, we analyze the runtime of the standard enumeration procedure on a ζ-reduced basis.

Theorem 3 *Let* $\mathbf{B} \in \mathbb{Z}^{m \times n}$ *be a* ζ*-reduced basis with an efficiently computable* $\zeta(i) \geq \sqrt{n}$ *for all* $i \in [n]$*. Then there is an efficiently computable set* $M \subset \mathbb{Z}^n$ *with* $|M| \leq 3^n \prod_{i=1}^n \zeta(i)$ *such that there is a vector* $\mathbf{x} \in M$ *with* $\|\mathbf{B}\mathbf{x}\| = \lambda_1(\mathbf{B})$.

Proof Let $\Delta = \det(\mathbf{B})$ and $r = \sqrt{n}\Delta^{1/n}$ be the Minkowski bound of $\mathscr{L}(\mathbf{B})$. We start out by noting that we can assume w.l.o.g. that $\|\mathbf{b}_i^*\| \leq \zeta(i)\Delta^{1/n}$ for all $i \in [n]$, because if there is an i with $\|\mathbf{b}_i^*\| > \zeta(i)\Delta^{1/n}$, we can ignore the entire sublattice $\mathscr{L}(\mathbf{B}_{[i,n]})$ due to ζ-reducedness and apply the result recursively to the reduced basis $\mathbf{B}_{[1,i-1]}$. Now we simply bound the number of steps in the enumeration by

$$|M| \leq \prod_{i=1}^{n} \left\lfloor \frac{2r}{\|\mathbf{b}_i^*\|} + 1 \right\rfloor. \tag{3.5}$$

Observe that for any real $\alpha \geq 0$ we have $\lfloor 2\alpha + 1 \rfloor \leq \max\{2, 3\alpha\}$. This can easily seen to be true: If $\alpha < 1$, then $2\alpha + 1 < 3$ and $\lfloor 2\alpha + 1 \rfloor \leq 2$. Otherwise, $\alpha \geq 1$ and $2\alpha + 1 \leq 2\alpha + \alpha = 3\alpha$. Setting $\alpha = r/\|\mathbf{b}_i^*\|$, we can bound each term in Equation 3.5

by

$$\left\lfloor \frac{2r}{\|\mathbf{b}_i^*\|} + 1 \right\rfloor \leq \max\{2, 3r/\|\mathbf{b}_i^*\|\} = \frac{r}{\|\mathbf{b}_i^*\|} \max\{2\|\mathbf{b}_i^*\|/r, 3\} \leq \frac{3\zeta(i)\Delta^{1/n}}{\|\mathbf{b}_i^*\|}$$

So, the size of M is at most $|M| \leq \prod_i \frac{3\zeta(i)\Delta^{1/n}}{\|\mathbf{b}_i^*\|} = 3^n \prod_i \zeta(i)$. □

Intuitively, Theorem 3 states that when calling the enumeration procedure on a ζ-reduced basis, the running time is bounded by the product of the ζ values for all basis vectors and a single exponential factor.

It is easy to see that an HKZ reduced basis is ζ-reduced for any constant function $\zeta(i) \geq \sqrt{n}$ by applying Minkowski's bound. The following lemma shows that the LLL algorithm computes ζ-reduced bases for an appropriate value of ζ.

Lemma 4 *Let* $\mathbf{B} \in \mathbb{Z}^{m \times n}$ *an LLL reduced basis. Then,* $\mathbf{B}$ *is* ζ*-reduced for* $\zeta(i) = 2^{\frac{n-1}{4}}$.

Proof Let $\mathbf{B}$ an LLL reduced basis of a lattice Λ. For any i, we prove that if $\lambda_1(\Lambda) \geq \lambda_1(\pi_{i-1}(\Lambda))$, then $\|\mathbf{b}_i^*\| \leq 2^{(n-1)/4} \det(\Lambda)^{1/n}$. Since $\mathbf{B}$ is LLL reduced, we have $\|\mathbf{b}_k^*\| \leq \sqrt{2}\|\mathbf{b}_{k+1}^*\|$ for all k. In particular, $\|\mathbf{b}_i^*\| \leq 2^{(k-i)/2}\|\mathbf{b}_k^*\|$ for all $k \geq i$, and $\|\mathbf{b}_1\| \geq \lambda_1(\Lambda) \geq \lambda_1(\pi_{i-1}(\Lambda)) \geq \min_{k \geq i} \|\mathbf{b}_k^*\| \geq \|\mathbf{b}_i^*\| 2^{-(n-i)/2}$. So, we also have $\|\mathbf{b}_i^*\| \leq 2^{(n-i+k)/2}\|\mathbf{b}_k^*\|$ for $k < i$. It follows that $\|\mathbf{b}_i^*\|^n \leq \prod_k 2^{((k-i) \bmod n)/2}\|\mathbf{b}_k^*\| = 2^{n(n-1)/4} \det(\Lambda)$ and $\|\mathbf{b}_i^*\| \leq 2^{(n-1)/4} \det(\Lambda)^{1/n}$. □

This immediately recovers the time complexity of the Fincke-Pohst algorithm [22], since it consists of preprocessing the basis using LLL and then performing the enumeration step. Using Lemma 4 and Theorem 3, this shows that asymptotically, Fincke-Pohst has a time complexity of $2^{n^2/4+O(n)}$.

The Preprocessing

In the following two subsections we consider two different types of preprocessing, which each can be viewed as generalizations of Kannan's algorithm [36] and Fincke-Pohst [22], respectively.

Kannan-style

The idea of Kannan's algorithm is easily explained: instead of just returning the shortest vector in the lattice, it HKZ reduces the basis $\mathbf{B}$ (which clearly solves SVP). In order to do so, it first uses alternating calls to LLL and itself recursively on $\pi_1(\mathbf{B})$ to quasi-HKZ reduce $\mathbf{B}$, then performs the enumeration step, and finally uses a recursive call to $\pi_1(\mathbf{B})$ to finalize the HKZ reduction. This produces a ζ-reduced basis before the enumeration with $\zeta(1) = 2\sqrt{n}$ and $\zeta(i) = \sqrt{n}$ for all $i > 1$. It follows that the complexity of the enumeration procedure on such bases (and by induction also the whole algorithm) is in $\tilde{O}(n^{n/2})$.

Note that the number of recursive calls of this algorithm, while asymptotically dominated by the final enumeration step, can be extremely large: Helfrich [35] proved that the number of top-level recursive calls is bounded by $O(\log n)$, leading to an overall multiplicative factor of $2^{O(n \log\log n)}$ in the complexity bound. The fact that heuristic arguments [33] indicate that the enumeration step should be significantly improved using stronger preprocessing suggests that it is this gigantic number of recursive calls that slows down this algorithm in practice to the extend that it is outperformed by the simple Fincke-Pohst algorithm for reasonable dimensions. In this section, we analyze to which extend one can reduce the number of recursive calls during the preprocessing while maintaining a $n^{O(n)}$ complexity bound. We remark, that we also show that it is possible to remove the recursive call during the postprocessing [52], but since this has little impact

on asymptotic or practical complexity, we refer the reader to [52] for details.

We begin by observing that in order to achieve the $\tilde{O}(n^{n/2})$ complexity, requiring $\zeta(1)$ to be this small seems to be an overkill. In fact, after running LLL we obtain $\zeta(1) = 2^{\frac{n-1}{4}}$. So, after a recursive call on the basis $\pi_1(\mathbf{B})$, which does not change $\zeta(1)$, we obtain a basis that is ζ-reduced for the function $\zeta(1) = 2^{\frac{n-1}{4}}$, $\zeta(i > 1) = \sqrt{n}$. Clearly, using this ζ function in Theorem 3 also exhibits a worst-case complexity of $\tilde{O}(n^{n/2})$. This *lightweight Kannan* algorithm reduces the number of recursive calls before enumeration from $\log(n)$ to 1 and so the overall number of recursive calls to 2^n, while preserving the asymptotic runtime.

Furthermore, our analysis suggests a natural generalization. We can introduce a degree of freedom by allowing a variable number $\kappa(n)$ of basis vectors to have an exponential ζ-bound as opposed to the sublinear bound obtained by the recursive call. This can be achieved by first LLL reducing the basis $\mathbf{B}$ and recursing on the basis $\pi_{\kappa(n)}(\mathbf{B})$ before enumerating. This will result in bounds of $\zeta(i \leq \kappa(n)) = 2^{\frac{n-1}{4}}$, $\zeta(i > \kappa(n)) = \sqrt{n}$, which, when plugged into Theorem 3, yield an upper bound of $\tilde{O}(2^{n\kappa(n)/4} n^{(n-\kappa(n))/2})$. Note that for $\kappa(n) = 1$ this variant corresponds to the lightweight Kannan algorithm, while for $\kappa(n) = n$ it degenerates to the Fincke-Pohst algorithm. Using intermediate values for $\kappa(n)$ allows us to interpolate between these two algorithms and thus this variant can be seen as a generalization of them. The new parameter can be used to balance the preprocessing with the enumeration: for larger values of $\kappa(n)$, the recursive call is cheaper, but enumeration is harder, and vice versa. In order to maintain an asymptotic upper bound of $n^{O(n)}$ we need to ensure that $\kappa(n) = O(\log(n))$. For example, $\kappa(n) \approx \log n$ results in a runtime of $\tilde{O}(n^{0.75n})$, only a little worse than the one for the lightweight Kannan of $\tilde{O}(n^{n/2})$, but with significantly lighter preprocessing.

Algorithm 3.3. Our HKZ Reduction Algorithm

`procedure` reduce ($\mathbf{B}$, κ)
Input: A lattice basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$, a functions $\kappa : [n] \to [n]$
Output: An HKZ reduced basis of $\mathscr{L}(\mathbf{B})$

1 $\mathbf{B} \leftarrow \text{LLL}(\mathbf{B})$
2 $\Delta \leftarrow \det(\mathbf{B})$, $k \leftarrow \kappa(n)$
3 `if` $\exists i \leq k : \|b_i^*\| > 2^{\frac{n-1}{2}} \Delta^{\frac{1}{n}}$ `then`
4 $\mathbf{v} \leftarrow \text{enum}(\mathbf{B}_{[1,i-1]})$
5 `return` $[\mathbf{v} | \pi_{\mathbf{v}}^{-1}(\text{reduce}(\pi_{\mathbf{v}}(\mathbf{B}), \kappa))]$
6 $\mathbf{B} \leftarrow [\mathbf{B}_{[1,k]} | \pi_k^{-1}(\text{reduce}(\pi_k(\mathbf{B}), \kappa)]$
7 $\mathbf{v} \leftarrow \text{enum}(\mathbf{B})$
8 `return` $[\mathbf{v} | \pi_{\mathbf{v}}^{-1}(\text{reduce}(\pi_{\mathbf{v}}(\mathbf{B}), \kappa))]$

Better Asymptotics for Non-Increasing Bases

In this section we will show that we can prove a tighter bound on the enumeration after preprocessing if we assume that the shape of the input basis is non-increasing, which is typically the case. What follows is an adaptation of the proof of Theorem 3 in [33], which can be found more explicitly in the extended version [34]. We show the result for $\kappa(n) = 1$, which yields the best asymptotic runtime in our previous analysis. The proof can easily be adapted to larger $\kappa(n)$ and yields the expected runtime.

We use the same starting point as the authors of [33], where it is proved that the number of nodes processed during a SVP enumeration with arbitrary bound r on an arbitrary basis $\mathbf{B}$ can be bounded by

$$2^{O(n)} \max_{I \subset [n]} \left(\frac{r^{|I|}}{\sqrt{n}^{|I|} \prod_{i \in I} \|\mathbf{b}_i^*\|} \right)$$

up to polynomial factors. Since our enumeration uses the bound $r = \sqrt{n} \det(\mathbf{B})^{1/n}$, this is equivalent to

$$2^{O(n)} \prod_I \frac{\det(\mathbf{B})^{1/n}}{\|\mathbf{b}_i^*\|}$$

where $I = \{i : \|\mathbf{b}_i^*\| < \det(\mathbf{B})^{1/n}\}$. Note that in the case of non-increasing bases, I is of

the form $I = [k, \ldots, n]$ for some $k \in [n]$. The following lemma shows the result:

Lemma 5 *Let* $\mathbf{B} \in \mathbb{Z}^{m \times n}$ *with* $\|\mathbf{b}_1\| \leq 2^{(n-1)/4} \det(\mathbf{B})^{1/n}$, $\pi_1(\mathbf{B})$ *be HKZ reduced, and* $\|\mathbf{b}_1\| \geq \|\mathbf{b}_2^*\| \geq \cdots \geq \|\mathbf{b}_n^*\|$. *Then for all* $k \in [n]$

$$\prod_{i=k}^{n} \frac{\det(\mathbf{B})^{1/n}}{\|\mathbf{b}_i^*\|} \leq 2^n n^{n/2e}.$$

Proof Let

$$\tilde{\Gamma}_n(k) = 2 \cdot \Gamma_{n-1}(k-1)$$

where $\Gamma_n(k) = \prod_{i=n-k}^{n-1} (\gamma_{i+1})^{1/2i}$ from Definition 2 of [33] and $\Gamma_n(0) = 1$. From Lemma 2 in [33] we immediately obtain $\tilde{\Gamma}_n(k) \leq 2\sqrt{n}^{\log\left(\frac{n}{n-k}\right)}$. Akin to the proof in the extended version of [33], we let $\pi_{[k,n]} = \prod_{i=k}^{n} \|\mathbf{b}_i^*\|^{1/(n-k+1)}$ and first prove the inequality

$$\pi_{[1,k]} \leq \tilde{\Gamma}_n(k)^{n/k} \pi_{[k+1,n]} \tag{3.6}$$

for all k by induction on k. For $k = 1$ the inequality follows from the assumption $\|\mathbf{b}_1\| \leq 2^{(n-1)/4} \det(\mathbf{B})^{1/n}$. The rest of the proof is identical to the one of Hanrot and Stehlé and shamelessly copied for completeness. Assume that the inequality holds for $k \geq 1$ and rewrite it as

$$\pi_{[1,k+1]}^{\frac{k+1}{k}} \cdot \|\mathbf{b}_{k+1}^*\|^{-\frac{1}{k}} \leq \tilde{\Gamma}_n(k)^{\frac{n}{k}} \cdot \pi_{[k+2,n]}^{\frac{n-k-1}{n-k}} \cdot \|\mathbf{b}_{k+1}^*\|^{\frac{1}{n-k}}$$

which is equivalent to

$$\pi_{[1,k+1]}^{\frac{k+1}{k}} \leq \tilde{\Gamma}_n(k)^{\frac{n}{k}} \cdot \pi_{[k+2,n]}^{\frac{n-k-1}{n-k}} \cdot \|\mathbf{b}_{k+1}^*\|^{\frac{n}{k(n-k)}}$$

From the HKZ reducedness assumption it follows that $\|\mathbf{b}_{k+1}^*\| \leq \sqrt{\gamma_{n-k}}^{\frac{n-k}{n-k-1}} \cdot \pi_{[k+2,n]}$,

which gives

$$\pi_{[1,k+1]}^{\frac{k+1}{k}} \leq \tilde{\Gamma}_n(k)^{\frac{n}{k}} \sqrt{\gamma_{n-k}}^{\frac{n}{k(n-k-1)}} \cdot \pi_{[k+2,n]}^{\frac{k+1}{k}} = \tilde{\Gamma}_n(k+1)^{\frac{n}{k}} \cdot \pi_{[k+2,n]}^{\frac{k+1}{k}}$$

which yields the induction step after raising to the power $k/(k+1)$.

From inequality (3.6) we obtain the inequality

$$\pi_{[k+1,n]} \geq \frac{\det(\mathbf{B})^{1/n}}{\tilde{\Gamma}_n(k)} \tag{3.7}$$

by raising (3.6) to the power of k/n, multiplying it with $\pi_{[k+1,n]}^{(n-k)/n}$, and using the identity $\det(\mathbf{B}) = \pi_{[1,k]}^{k} \cdot \pi_{[k+1,n]}^{n-k}$. From (3.7) we get

$$\left(\frac{\det(\mathbf{B})^{1/n}}{\pi_{[k+1,n]}}\right)^{n-k} \leq \tilde{\Gamma}_n(k)^{n-k} \leq 2^{n-k}\sqrt{n}^{(n-k)\cdot\log\left(\frac{n}{n-k}\right)} \leq 2^n \sqrt{n}^{n/e}. \qquad \square$$

A Variant Based on Dual HKZ Reduction

As mentioned, in [33] the authors were able to improve the analysis of Kannan's algorithm to obtain a worst case bound on the asymptotic running time of $\tilde{O}(n^{n/2e})$ instead of Helfrich's $\tilde{O}(n^{n/2})$ [35]. It is unclear if and how the techniques from [33] can be applied to our algorithm to achieve a similar bound (with exception of the special case in the previous subsection). Here we present a variant of our algorithm, for which we can rigorously prove the same complexity bound as in [33]. In [52] we presented experimental results, which show that in practice this variant has efficiency comparable to the previous algorithm.

This variant, presented in Algorithm 3.4, maintains the idea of restricting recursive

Algorithm 3.4. Variant of our HKZ Reduction Algorithm

`procedure` reduce ($\mathbf{B}$, κ)
Input: A lattice basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$, a function $\kappa : [n] \to [n]$
Output: An HKZ reduced basis of $\mathscr{L}(\mathbf{B})$

```
1   B ← LLL(B)
2   do
3     B ← [B_[1,k] | π_k^{-1}(reduce(π_k(B), κ))]
4     V ← reduce(B_[1,k-1], κ)
5     if ‖v_1‖ ≤ ‖b_k^*‖
6       return [v_1 | π_{v_1}^{-1}(reduce(π_{v_1}(B), κ))]
7     B ← [dualHKZ(B_[1,k]) | B_[k+1,n]]
8   while change occurred
9   v ← enum(B)
10  return [v | π_v^{-1}(reduce(π_v(B), κ))]
```

calls to dimension $n-k$, but potentially makes logarithmically (in n) many of them interleaved with calls to a k dimensional dual HKZ reduction, where $k = \kappa(n)$. The dual HKZ reduction can be realized by the techniques outlined in Section 3.1.

We first bound the enumeration step and then the number of loop iterations during the preprocessing in Algorithm 3.4 .

Lemma 6 *Let $\mathbf{B} \in \mathbb{Z}^{m \times n}$ be such that $\mathbf{B}_{[1,k]}$ is dual HKZ reduced and $\pi_{k-1}(\mathbf{B})$ is HKZ reduced for some $k \in [n]$. The shortest vector vector in $\mathscr{L}(\mathbf{B})$ can be computed by enumeration while exploring at most $k^{O(n)} n^{(n-k)/2e+o(n)}$ nodes.*

Proof Let $r_k = \sqrt{k}\det(\mathscr{L}(\mathbf{B}_{[1,k]}))^{1/k}$. Note that by Minkowski's theorem $r_k \geq \lambda_1(\mathscr{L}(\mathbf{B}_{[1,\kappa(n)]})) \geq \lambda_1(\mathscr{L}(\mathbf{B}))$, so $\mathscr{L}(\mathbf{B})$ contains a vector of length at most r_k. Furthermore, note that the shortest vector can be found by enumerating all vectors of length r_k in $\pi_k(\mathscr{L}(\mathbf{B}))$, followed by a CVP computation for each of them. We first bound the number of nodes explored during the first step. Note that due to the dual HKZ reducedness of $\mathbf{B}_{[1,k]}$ we have $1/\|\mathbf{b}_k^*\| \leq \sqrt{k}/\det(\mathscr{L}(\mathbf{B}_{[1,k]}))^{1/k}$, and so $r_k \leq k\|\mathbf{b}_k^*\|$. By Theorem 3 in [33], the number of nodes explored during the first step can be bounded by $k^{n-k} n^{(n-k)/2e+o(n)}$. For each vector found, the CVP problem can be solved exploring

at most $k! = k^{O(k)}$ nodes due to the dual HKZ reducedness of $\mathbf{B}_{[1,k]}$ [8]. Multiplying the two bounds gives the result. □

Lemma 7 *Let $\mathbf{B} \in \mathbb{Z}^{m \times n}$ be LLL reduced. Then the number of iterations between Line 3 and 7 is at most logarithmic in $k \cdot n$ with base $\alpha = k/(k-1)$.*

Proof The proof is a generalization of Helfrich's proof of the number of loop iterations performed by Kannan's algorithm ([35], Lemma 3.3). Let $\lambda_1 = \lambda_1(\mathscr{L}(\mathbf{B}))$ and $\bar{\mathbf{B}}$ be such that it solves the densest $(k-1)$-sublattice problem and $\bar{\mathbf{B}}_{[1,k-1]}$ is LLL reduced. We need the following facts:

Fact 12 *If $\mathbf{B} \in \mathbb{Z}^{m \times k}$ is dual HKZ reduced, then*

$$\det(\mathscr{L}(\mathbf{B}_{[1,k-1]})) \leq \sqrt{\gamma_k} \det(\mathscr{L}(\mathbf{B}))^{(k-1)/k}.$$

Proof Can easily be shown by applying Minkowski's theorem to the dual of $\mathbf{B}$. □

Fact 13 *When executing Line 7 of Algorithm 3.4, $\|\mathbf{b}_k^*\| \leq \lambda_1$ holds.*

Proof Let $\mathbf{v} \in \mathscr{L}(\mathbf{B})$ with $\|\mathbf{v}\| = \lambda_1$. If $\pi_{k-1}(\mathbf{v}) > 0$, the fact follows from the HKZ reduction step. Otherwise, $\mathbf{v} \in \mathscr{L}(\mathbf{B}_{[1,k-1]})$ and the fact easily follows from the check in Line 5. □

Fact 14 *If $\mathbf{B}$ is LLL reduced, then* $\det(\mathscr{L}(\mathbf{B}_{[1,k-1]}))/\det(\mathscr{L}(\bar{\mathbf{B}}_{[1,k-1]})) \leq 2^{O(kn)}$.

Proof W.l.o.g. we can assume that $\|\mathbf{b}_i^*\| \leq 2^{(n-i)/2}\|\mathbf{b}_1\| \leq 2^n \lambda_1$, because otherwise $\lambda_1(\pi_{i-1}(\mathscr{L}(\mathbf{B}))) > \|\mathbf{b}_1\|$ (for brevity we ignored an explicit check in the presentation of Algorithm 3.4). It follows that $\det(\mathscr{L}(\mathbf{B}_{[1,k-1]})) \leq 2^{kn}\lambda_1^{k-1}$. From the properties of LLL

reduction we get $\lambda_1 \leq \bar{\mathbf{b}}_1 \leq 2^{(k-2)/4}\det(\mathscr{L}(\bar{\mathbf{B}}_{[1,k-1]}))^{1/(k-1)}$ and so $\det(\mathscr{L}(\bar{\mathbf{B}}_{[1,k-1]})) \geq \lambda_1^{k-1}/2^{(kn)/4}$. Putting the two inequalities together proves the fact. $\square$

We define $r(\mathbf{B}) = \det(\mathbf{B}_{[1,k-1]})/\det(\bar{\mathbf{B}}_{[1,k-1]})$. Note that the HKZ reduction step in Line 3 does not change the value of $r(\mathbf{B})$. Now let $\mathbf{B}$ be the matrix during some iteration of the loop before the dual HKZ reduction step (Line 7) and $\mathbf{B}'$ its result. Then we have

$$
\begin{aligned}
r(\mathbf{B}') &= \frac{\det(\mathbf{B}'_{[1,k-1]})}{\det(\bar{\mathbf{B}}_{[1,k-1]})} \\
&\leq \sqrt{\gamma_k}\frac{\det(\mathbf{B}'_{[1,k]})^{(k-1)/k}}{\det(\bar{\mathbf{B}}_{[1,k-1]})} && \textit{Fact 12} \\
&\leq \sqrt{\gamma_k}\frac{\det(\mathbf{B}_{[1,k]})^{(k-1)/k}}{\det(\bar{\mathbf{B}}_{[1,k-1]})} \\
&\leq \sqrt{\gamma_k}r(\mathbf{B})^{(k-1)/k}\frac{\|\mathbf{b}_k^*\|^{(k-1)/k}}{\det(\bar{\mathbf{B}}_{[1,k-1]})^{1/k}} \\
&\leq \sqrt{\gamma_k}r(\mathbf{B})^{(k-1)/k}\frac{\lambda_1^{(k-1)/k}}{\det(\bar{\mathbf{B}}_{[1,k-1]})^{1/k}} && \textit{Fact 13} \\
&\leq \sqrt{\gamma_k}r(\mathbf{B})^{(k-1)/k}\frac{\lambda_1(\mathscr{L}(\bar{\mathbf{B}}))^{(k-1)/k}}{\det(\bar{\mathbf{B}}_{[1,k-1]})^{1/k}} \\
&\leq \sqrt{\gamma_k}r(\mathbf{B})^{(k-1)/k}\sqrt{\gamma_{k-1}}^{(k-1)/k} \\
&\leq \gamma_k r(\mathbf{B})^{1/\alpha}
\end{aligned}
$$

Now denote the value of $r(\mathbf{B})$ after the i-th iteration by r_i. We have just shown that $r_{i+1} \leq \gamma_k r_i^{1/\alpha}$ and it follows that $r_i \leq \gamma_k^i r_1^{1/\alpha^i} \leq \gamma_k^i 2^{O(kn)/\alpha^i}$ by Fact 14. So after $i = \log_\alpha O(kn)$ iterations $r_i \leq 2 \cdot O(kn)^{\log_\alpha \gamma_k}$. Introducing a slack $\delta > 1$ for the dual HKZ reduction step (which does not affect the bound given in Lemma 6), there are at most $\log_\alpha(\gamma_k)\log_\delta(O(kn))$ more iterations. $\square$

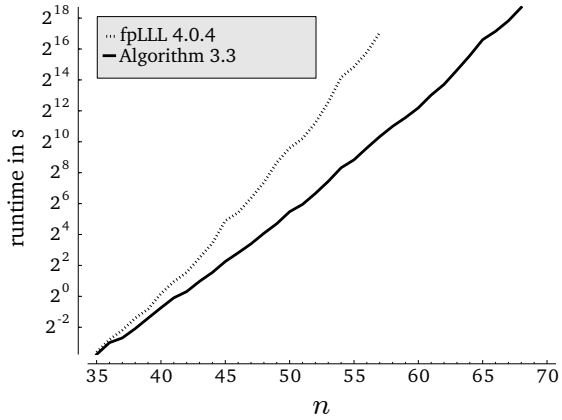


Figure 3.1. Comparison of the runtime of original `fpLLL` and Algorithm 3.3 ($\kappa = \log n$)

Application to Cryptanalysis

In [52] we gave details for an experimental analysis of our algorithm, showing that $\kappa(n) = \log(n)$ seems to be a good choice. As an exemplary result, we show in Figure 3.1 a comparison of our algorithm with a then state-of-the-art implementation of the FinckePohst algorithm from `fpLLL`. It shows that the running time of FinckePohst starts to increase more rapidly beyond dimension 30 and there seems to be a qualitative difference between the running times of the algorithms, with the FinckePohst algorithm exhibiting a clear superexponential slowdown, and the graph for our new algorithm much closer to a straight line (corresponding to an almost linear exponent in the running time, i.e. $2^{O(n \log n)}$).

We emphasize that our experimental analysis of the algorithm is only preliminary as the tested dimensions are fairly low compared to what is possible with today's state of the art (see, e.g. the SVP challenge). Nonetheless, to get a sense of how well our algorithm performs in higher dimensions, we used standard statistical methods to fit curves to the data collected in our experiments. To determine the practical complexity of the enumeration on reduced bases, we selected a model based on our theoretical analysis, $f(n) = 2^{c_1 n^2} \cdot n^{c_2 n} \cdot 2^{c_3 n}$, and fitted it to the data collected only during the top

Table 3.1. Parameters of model $2^{c_1 n^2} \cdot n^{c_2 n} \cdot 2^{c_3 n}$ after curve fitting to top level enumeration

Algorithm	c_1	c_2	c_3
FinckePohst	0.0068	0.0000	0.3195
Ours	0.0009	0.0839	0.0508

Table 3.2. Parameters of models f_1 (for FinckePohst) and f_2 (for our algorithm) after curve fitting

	$f_1(n) = 2^{a_1 n^2 + a_2 n}$		$f_2(n) = n^{b_1 n} \cdot 2^{b_2 n}$	
Preprocessing	a_1	a_2	b_1	b_2
LLL	0.0045	0.4469	0.0280	0.4389
BKZ-10	0.0024	0.4830	0.0245	0.4504
BKZ-20	0.0019	0.4962	0.0237	0.4518
BKZ-30	0.0016	0.5028	0.0225	0.4574

level enumeration, i.e. ignoring pre- and postprocessing. The resulting constants for FinckePohst and our algorithm are shown in Table 3.1. For FinckePohst, the running time (in low dimension) is dominated by the single exponential component $2^{c_3 n}$, but there is also a quadratic term $2^{c_1 n^2}$ which becomes dominant in sufficiently high dimension. (Notice also the complete absence of a quasilinear exponent $2^{c_2 n \log n}$.) So, the runtime of FinckePohst is clearly $2^{O(n^2)}$ also in practice when the dimension is sufficiently large. On the other hand, for our algorithm the dominating factor is the quasilinear exponent associated to c_2. Apart from the small constant c_1 in the case of our algorithm, which we attribute to noise, this is consistent with the theoretical analysis. For further exploration, we selected the model $f_1(n) = 2^{a_1 n^2 + a_2 n}$ for the FinckePohst algorithm and $f_2(n) = n^{b_1 n} \cdot 2^{b_2 n}$ for our algorithm and fitted them to the entire runtimes, i.e. with pre- and postprocessing, including variants where we replaced LLL preprocessing with BKZ for both algorithms. The result is shown in Table 3.2. The table demonstrates that the improvement due to BKZ decreases with increasing blocksize.

In [25] the authors claim that enumeration can be sped up heuristically by a factor

of $2^{n/2}$ with extreme pruning. Recall that the basic idea of extreme pruning is to heavily cut down on the enumeration and compensate the sacrificed success probability by a large number of repetitions with randomized inputs. FinckePohst is an ideal candidate for extreme pruning due to the cheap preprocessing, which has to be applied in each iteration. After introducing the corresponding speed-up into f_1 we can estimate the dimension n at which our algorithm becomes more efficient than FinckePohst with extreme pruning by computing the cross-over point between (the modified) f_1 and f_2. This point is reached for $n = 155$, which seems already very close to practically tractable dimensions. It is not immediately clear if extreme pruning can be applied to our algorithm due to its heavier preprocessing. However, in the same work the authors of [25] show that a speed-up of $2^{n/4}$ can be achieved by non-extreme pruning, where the enumeration is pruned but only sacrificing very little success probability. This non-extreme form of pruning can readily be applied to our algorithm and we expect it to result in similar speed-ups. If this is indeed the case the cross-over point at which our algorithm becomes more efficient than FinckePohst with extreme pruning would drop to $n = 95$ – well below the limit of today's tractability.

Applying the same approach to variants that use BKZ and extreme pruning is a little problematic, as extreme pruning is likely to suffer from heavy preprocessing. The precise impact of BKZ on the practical complexity of extreme pruning has to the best of our knowledge not been investigated in detail. However, the authors of [25] give a prediction for the number of nodes that have to be enumerated by extreme pruning in dimension $n = 110$ using BKZ-32 preprocessing, which is $2.5 \cdot 10^{13}$. Using our model, we expect the number of nodes to be enumerated by our algorithm with BKZ-30 and (non-extreme) pruning to be about $8.4 \cdot 10^{11}$. This already corresponds to a speed-up of about 30 even without considering the overhead incurred by the repeated application of BKZ-32 necessary for extreme pruning. We expect this speed-up factor to increase

rapidly in larger dimensions due to the superior asymptotics of our algorithm.

We can use our model to estimate at which point the asymptotically superior sieving becomes more efficient. Even though sieving algorithms have the drawback of exponential memory requirements, we will ignore this fact and focus on the runtime. In [39] the running time of HashSieve, a very recent sieving variant and to the best of our knowledge the most efficient sieving algorithm in practice to date, was estimated to be $2^{0.45n-19}$ seconds in dimension n in practice. In order to compare our algorithm meaningfully to this estimate, we need to convert the number of nodes calculated by f_2 to computing time on a comparable computer that was used in [39]. For this we use the observation made in [25] that an efficient implementation of enumeration can process a node in approximately 200 clock cycles. Computing the intersection of f_2 with the estimate for sieving of $2^{0.45n-19}$ suggests that sieving is more efficient than our algorithm starting already in dimension $n = 35$. However, if introducing the potential speed-up of $2^{n/4}$ to f_2 due to pruning, this cross-over point rises to $n = 745$. Using for f_2 the parameters resulting from the application of BKZ-30 to the basis before running our algorithm (cf. Table 3.2) and including the pruning heuristic, we expect our algorithm to be more efficient up to dimension $n = 1895$. The latter two values for n are both by far out of reach for today's state of the art. We remark that the estimate from [39] does not take block reduction as preprocessing into account. This is likely to improve HashSieve somewhat, but we do not believe that it will change the picture significantly.

Remark about FinckePohst and Sieving Although not directly relevant to the algorithm proposed in this work, we can use the model for FinckePohst and compare it to sieving. Without pruning, FinckePohst is more efficient than sieving up to dimension $n = 31$, with non-extreme pruning up to $n = 71$ and with extreme pruning up to $n = 120$. This is consistent with results of the SVP challenge, which gives us confidence in our

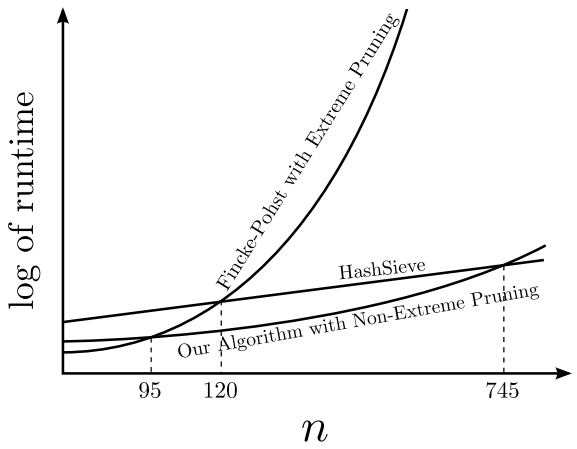


Figure 3.2. Diagram of estimated runtimes and cross-over points (not to scale). All Algorithms use only LLL preprocessing.

model. Furthermore, these numbers suggest that in the search for fast practical algorithms, in particular in dimensions relevant to currently unbroken challenges, one should focus on (non-extremely) pruned enumeration or sieving rather than extreme pruning.

An illustration of the cross-over points is given in Figure 3.2. We reiterate that this analysis is preliminary and more experimental evidence is necessary to support our hypothesis.

Block Reduction

We now turn to another type of preprocessing, typically encountered in practical implementations of enumeration. Here we use FinckePohst with its LLL preprocessing as a starting point. Recall that LLL can be considered a special case of block reduction algorithms with block size 2. In order to increase the performance of the algorithm, one typically increases the time spent on preprocessing by increasing the block size parameter. On an intuitive level, this will improve the shape of the basis and thus improve the complexity of the enumeration step. In the following we analyze for the first time the impact of such a preprocessing on the asymptotic complexity of the enumeration step, using our framework of ζ-reduction.

Consider an algorithm that reduces an input basis $\mathbf{B}$ using a block reduction algorithm with parameter k and then runs enumeration to find the shortest vector. We are interested in the complexity of the enumeration step depending on the parameter k. We first briefly discuss the corner cases. For $k = 2$, the algorithm reduces to FinckePohst, which we already analyzed in Section 3.2. Now consider the algorithm with parameter $k = n - 1$. In this case, it has a striking similarity to Kannan's algorithm: note that Kannan's algorithm can be viewed as alternately calling a SVP oracle on $\mathbf{B}_{[1,2]}$ and a HKZ oracle (instantiated with a recursive call) on $\pi_1(\mathbf{B})$ in order to achieve quasi-HKZ reducedness. The only difference between this kind of preprocessing and, for example, BKZ-$(n-1)$ is that the latter alternates between calling an SVP oracle on $\mathbf{B}_{[1,n-1]}$ and a HKZ oracle on $\pi_1(\mathbf{B})$. So a BKZ-$(n-1)$ reduced basis is also quasi-HKZ reduced and thus strictly stronger reduced than after Kannan's preprocessing. In particular, the enumeration complexity is less than the one for Kannan's algorithm, which we know to be $O(n^{n/2e})$.

We first focus on the case of BKZ preprocessing and analyze the ζ bounds that it achieves depending on the block size parameter in the following lemma.

Lemma 8 *If $\mathbf{B} \in \mathbb{Z}^{m \times n}$ is BKZ-k reduced then it is ζ-reduced with $\zeta(i) = k^{\frac{n-1}{2(k-1)}+\frac{3}{2}}$.*

Proof We prove the contrapositive and assume $\lambda_1(\pi_{i-1}(\mathbf{B})) \leq \lambda_1(\mathbf{B})$. Since $\pi_{i-1}(\mathbf{B})$ and $\mathbf{B}_{[1,i-1]}$ are BKZ-k reduced, we have

$$\begin{aligned}
\|\mathbf{b}_i^*\| &\leq k^{\frac{n-i}{(k-1)}} \lambda_1(\pi_{i-1}(\mathbf{B})) \\
&\leq k^{\frac{n-i}{(k-1)}} \lambda_1(\mathbf{B}) \\
&\leq k^{\frac{n-i}{(k-1)}} \|\mathbf{b}_1\| \\
&\leq k^{\frac{n-i}{(k-1)}} k^{\frac{i-2}{2(k-1)}+\frac{3}{2}} \det(\mathbf{B}_{[1,i-1]})^{1/(i-1)}
\end{aligned}$$

and so

$$\|\mathbf{b}_i^*\|^{i-1} \leq k^{\frac{(i-1)(n-i)}{(k-1)}+\frac{(i-1)(i-2)}{2(k-1)}+\frac{3}{2}(i-1)} \det(\mathbf{B}_{[1,i-1]})$$

By (2.3) we also have $\|\mathbf{b}_i^*\|^{n-i+1} \leq k^{\frac{(n-i)(n-i+1)}{2(k-1)}+\frac{3}{2}(n-i+1)} \det(\pi_{i-1}(\mathbf{B}))$. Multiplying those two bounds and doing some arithmetic gives

$$\|\mathbf{b}_i^*\|^n \leq k^{\frac{(i-1)(n-i)}{(k-1)}+\frac{(i-1)(i-2)}{2(k-1)}+\frac{(n-i)(n-i+1)}{2(k-1)}+\frac{3}{2}n} \det(\mathbf{B}) \leq k^{\frac{n(n-1)}{2(k-1)}+\frac{3}{2}n} \det(\mathbf{B})$$

□

Using Theorem 3 we can easily deduce a runtime bound for the enumeration step.

Corollary 1 *Given a BKZ-k reduced basis* $\mathbf{B} \in \mathbb{Z}^{m \times n}$*, enumeration can solve SVP in* $\Lambda(\mathbf{B})$ *in* $k^{\frac{n(n-1)}{2(k-1)}+\frac{3}{2}n} 2^{O(n)}$.

For $k = 2$ (FinckePohst) and $k = n-1$ ($\approx$ Kannan) we get the expected bounds up to constants in the exponent. Other values for k interpolate the two algorithms offering an improvement in the exponent of the dominating factor of FinckePohst of about $\log(k)/k$. Up to constants in the exponent, this proves that the enumeration step after BKZ is as efficient as after Kannan's preprocessing as long as $k = O(n)$.

We continue by also showing how to apply ζ-reduction to slide reduced bases, which leads to slightly improved results.

Lemma 9 *If* $\mathbf{B} \in \mathbb{Z}^{m \times n}$ *is slide reduced with parameter k, the t-th projected block* $\pi_{tk}(\mathbf{B}_{[tk+1,(t+1)k]})$ *is* ζ*-reduced for*

$$\zeta(tk < i \leq (t+1)k) = k^{\frac{n-1}{2(k-1)}+\frac{k}{k-1}-\frac{tk^2}{n(k-1)}}$$

Proof As before, we assume $\lambda_1(\pi_{i-1}(\mathbf{B})) \leq \lambda_1(\mathbf{B})$. We start by showing the lemma for the first vector $\|\mathbf{b}_i^*\|$ of the block. In this case $\mathbf{B}_{[1,i-1]}$ and $\pi_{i-1}(\mathbf{B})$ are also slide reduced and we can apply the same approach as for BKZ:

$$\begin{aligned}\|\mathbf{b}_i^*\| \leq & k^{\frac{n-i-k}{(k-1)}} \lambda_1(\pi_{i-1}(\mathbf{B})) \\ \leq & k^{\frac{n-i-k}{(k-1)}} \lambda_1(\mathbf{B}) \\ \leq & k^{\frac{n-i-k}{(k-1)}} \|\mathbf{b}_1\| \\ \leq & k^{\frac{n-i-k}{(k-1)}} k^{\frac{i-2}{2(k-1)}} \det(\mathbf{B}_{[1,i-1]})^{1/(i-1)}\end{aligned}$$

and so

$$\|\mathbf{b}_i^*\|^{i-1} \leq k^{\frac{(i-1)(n-i-k)}{(k-1)} + \frac{(i-1)(i-2)}{2(k-1)}} \det(\mathbf{B}_{[1,i-1]})$$

By (2.5) we also have $\|\mathbf{b}_i^*\|^{n-i+1} \leq k^{\frac{(n-i)(n-i+1)}{2(k-1)}} \det(\pi_{i-1}(\mathbf{B}))$. Again, multiplying those two bounds gives

$$\begin{aligned}\|\mathbf{b}_i^*\|^n &\leq k^{\frac{(i-1)(n-i-k)}{(k-1)} + \frac{(i-1)(i-2)}{2(k-1)} + \frac{(n-i)(n-i+1)}{2(k-1)}} \det(\mathbf{B}) \\ &\leq k^{\frac{n(n-1)-2k(i-1)}{2(k-1)}} \det(\mathbf{B})\end{aligned} \tag{3.8}$$

which implies $\|\mathbf{b}_i^*\| \leq k^{\frac{n-1}{2(k-1)}} \det(\mathbf{B})^{1/n}$ and shows the result for the first vector of each block, because $\frac{k}{k-1} \geq \frac{tk^2}{n(k-1)}$ and so $\zeta(i) \geq k^{\frac{n-1}{2(k-1)}}$.

We now generalize to arbitrary i. Let $j = (t+1)k$, i.e. the end of the block. If $\lambda_1(\pi_j(\mathbf{B})) > \lambda_1(\pi_{i-1}(\mathbf{B}))$ then the shortest vector in $\pi_{i-1}(\mathbf{B})$ is in $\pi_{i-1}(\mathbf{B}_{[i,j]})$ and $\|\mathbf{b}_i^*\| = \lambda_1(\pi_{i-1}(\mathbf{B}))$, because $\pi_{i-1}(\mathbf{B}_{[i,j]})$ is HKZ reduced. It follows that $\|\mathbf{b}_i^*\|$ is ζ-reduced for all $\zeta(i) \geq \sqrt{n} \leq k^{\frac{n-1}{2(k-1)}}$. Now let $\lambda_1(\pi_j(\mathbf{B})) \leq \lambda_1(\pi_{i-1}(\mathbf{B}))$. Then by assumption $\lambda_1(\pi_j(\mathbf{B})) \leq \lambda_1(\pi_{i-1}(\mathbf{B})) \leq \lambda_1(\mathbf{B})$, so (3.8) holds for $\mathbf{b}_{j+1}^*$. Utilizing the fact that $\pi_{i-1}(\mathbf{B}_{[i,j]})$ is HKZ reduced and $\pi_i(\mathbf{B}_{[i+1,j+1]})$ is DSVP reduced, we easily deduce by Minkowski's theorem that $\|\mathbf{b}_i^*\| \leq \kappa^{\frac{\kappa}{\kappa-1}} \|\mathbf{b}_{j+1}^*\|$ where $\kappa = j - i + 1$. Putting this and

(3.8) together, we get:

$$\begin{aligned}\|\mathbf{b}_i^*\| \leq & \kappa^{\frac{\kappa}{\kappa-1}} \|\mathbf{b}_{j+1}^*\| \\ \leq & k^{\frac{k}{k-1}} k^{\frac{n(n-1)-2kj}{2n(k-1)}} \det(\mathbf{B})^{1/n} \\ \leq & k^{\frac{n-1}{2(k-1)}+\frac{k}{k-1}-\frac{tk^2}{n(k-1)}} \det(\mathbf{B})^{1/n}\end{aligned}$$

□

Again, using Theorem 3, we obtain a bound on the runtime of enumeration on slide reduced bases.

Corollary 2 *Given a k-slide reduced basis* $\mathbf{B} \in \mathbb{Z}^{m \times n}$*, enumeration can solve the SVP in* $\Lambda(\mathbf{B})$ *in* $k^{\frac{n(n-1)}{2(k-1)}+\frac{(n-k)}{2}} 2^{O(n)}$.

Proof The corollary follows from a short sequence of equations:

$$\begin{aligned}\prod_{t=1}^{n/k} \zeta((t-1)k+1)^k = & k^{\frac{n(n-1)}{2(k-1)}+\frac{nk}{k-1}-\frac{k^3}{n(k-1)}\sum_{t=1}^{n/k} t} \\ = & k^{\frac{n(n-1)}{2(k-1)}+\frac{nk}{k-1}-\frac{nk+k^2}{2(k-1)}} \\ = & k^{\frac{n(n-1)}{2(k-1)}+\frac{k(n-k)}{2(k-1)}} \approx k^{\frac{n(n-1)}{2(k-1)}+\frac{(n-k)}{2}}\end{aligned}$$

□

Not surprisingly, due to the better bounds achieved on $\|\mathbf{b}_1^*\|$, slide reduction yields a stronger ζ-reduction and thus improves the bound on the enumeration. However, plugging $k = n-1$ into the bound[2] shows that the bound is still worse than the one for

[2]Technically, this choice of parameter is not possible for slide reduction as it requires $k|n$. But plugging in this value should give a good estimation of how tight the bound is by comparison with Kannan's algorithm.

Kannan, but only by a factor $1/e$. We leave it as an interesting open question if one can achieve such a bound for block reduced bases.

Remark Recall that block reduction algorithms use a SVP oracle in dimension k. Obviously, we can use recursive calls to our enumeration algorithm (including block reduction) to implement this oracle. In the case of BKZ we can use the slightly worse bound obtained in [32] instead of Equation (2.2). This will give us worse constants in the exponents, but has the advantage that the number of (top level) recursive calls during the preprocessing is polynomially bounded, which bounds the overall number of recursive calls by $n^{O(n)}$. This proves that using the algorithm proposed in [32] combined with ζ-reduction, SVP can be solved by block reduction and enumeration in $n^{O(n)}$ steps by setting $k = \omega(n)$. Alternatively, we can use slide reduction instead of BKZ to achieve a similar result. To the best of our knowledge, such a bound was only known for Kannan's algorithm and the variant proposed in Section 3.3.1, up to this point. The impact of using this asymptotic model for extrapolation in cryptanalysis has been extensively studied in [2].

Chapter 3, in full, is a combination of material as it appears (with minor modifications) in the papers

- "Fast Lattice Point Enumeration with Minimal Overhead" [52] by Daniele Micciancio and Michael Walter, published in the proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016). The dissertation author was the primary investigator and author of this paper.

- "Lattice Point Enumeration on Block Reduced Bases" [79] by Michael Walter, published in the proceedings of the Eighth International Conference on Information Theoretic Security (ICITS 2015). The dissertation author is the sole author of this paper.

- “Practical, Predictable Lattice Basis Reduction” [54] by Daniele Micciancio and Michael Walter, published in the proceedings of the Thirty-Fifth Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2016). The dissertation author was the primary investigator and author of this paper.

Chapter 4

Lattice Block Reduction

The main topic of this chapter is lattice reduction. Recall from Section 2.4.2 two of the state-of-the-art block reduction algorithms: BKZ and Slide reduction. While Slide reduction has a very clean analysis and is the theoretically best block reduction algorithm to date, BKZ is the de facto standard for cryptanalysis due to its superior performance in practice. Unfortunately, the behavior of BKZ is hard to predict, which is a major issue for cryptanalysis. While progress has been made with regards to its running time [32] after 20 years of research effort, predicting its output quality is still tricky, since the best strategies [16] are cumbersome and build on questionable assumptions. One of the main sources of problems is the fact that BKZ even with large block size relies on SVP oracles in low dimensions and as stated in Section 2.4.3, the Gaussian heuristic cannot be expected to hold in such low dimensions, so additional assumptions need to be made.

In this section we introduce a block reduction algorithm that is competitive in practice, but matches the theoretical bounds of Slide reduction and allows for precise predictions solely based on the Gaussian heuristic.

Self-Dual BKZ

Like BKZ our new algorithm is parameterized by a block size k and a SVP oracle in dimension k, and acts on the input basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$ by iterating tours. The beginning of

every tour is exactly like a BKZ tour, i.e. SVP reducing every block $\pi_i(\mathbf{B}_{[i,i+k-1]})$ from $i = 1$ to $n-k+1$. We will call this part a *forward tour*. For the last block, which BKZ simply HKZ reduces and where most of the problems for meaningful predictions stem from, we do something different. Instead, we dual SVP the last block and proceed by dual SVP reducing all blocks of size k backwards (which is a *backward tour*). After iterating this process (which we call a *tour* of Self-Dual BKZ) the algorithm terminates when no more progress is made. The algorithm is formally described in Algorithm 4.1.

Algorithm 4.1. Self-Dual BKZ

`procedure` DBKZ ($\mathbf{B}$, k, SVP_k)

Input: A lattice basis $\mathbf{B} \in \mathbb{Z}^{m \times n}$, a block size k, a SVP oracle in dimension k

Output: A k-reduced basis $\mathbf{B}'$ (See Definition 23 for a formal definition.)

1 `do`
2 `for` $i = 1 \ldots n-k$
3 SVP reduce $\pi_i(\mathbf{B}_{[i,i+k-1]})$ using SVP_k
4 `for` $i = n-k+1 \ldots 1$
5 dual SVP reduce $\pi_i(\mathbf{B}_{[i,i+k-1]})$ using SVP_k
6 `while` *progress is made*
7 `return` $\mathbf{B}$

Note that, like BKZ, Self-Dual BKZ (DBKZ) is a proper block generalization of the LLL algorithm, which corresponds to the case $k = 2$.

The terminating condition in Line 6 is left ambiguous at this point on purpose as there are several sensible ways to approach this as we will see in the next section. One has to be careful to, on the one hand guarantee termination, while on the other hand achieving a meaningful reducedness definition.

Reducedness Definition

The output of Algorithm 4.1 satisfies the following reducedness definition upon termination:

Definition 23 *A basis* $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ *is k-reduced if either* $n < k$*, or it satisfies the following conditions:*

- $\|\mathbf{b}_k^*\|^{-1} = \lambda_1(\widehat{\mathscr{L}(\mathbf{B}_{[1,k]})})$*, and*
- *for some SVP reduced basis* $\tilde{\mathbf{B}}$ *of* $\mathscr{L}(\mathbf{B}_{[1,k]})$*,* $\pi_2([\tilde{\mathbf{B}}|\mathbf{B}_{[k+1,n]}])$ *is k-reduced.*

We first prove that Algorithm 4.1 indeed achieves Definition 23 when used with a specific terminating condition:

Lemma 10 *Let* $\mathbf{B}$ *be an n-dimensional basis. If* $\pi_{k+1}(\mathbf{B})$ *is the same before and after one loop of Algorithm 4.1, then* $\mathbf{B}$ *is k-reduced.*

Proof The proof is inductive: for $n = k$ the result is trivially true. So, assume $n > k$, and that the result already holds for $n-1$. At the end of each iteration, the first block $\mathbf{B}_{[1,k]}$ is dual-SVP reduced by construction. So, we only need to verify that for some $\tilde{\mathbf{B}}$ an SVP reduced basis for $\mathscr{L}(\mathbf{B}_{[1,k]})$, the projection $\pi_2([\tilde{\mathbf{B}}|\mathbf{B}_{[k+1,n]}])$ is also k-reduced. Let $\tilde{\mathbf{B}}$ be the SVP reduced basis produced in the first step. Note that the first and last operation in the loop do not change $\mathscr{L}(\mathbf{B}_{[1,k]})$ and $\mathbf{B}_{[k+1,n]}$. It follows that $\pi_{k+1}(\mathbf{B})$ is the same before and after the partial tour (the tour without the first and the last step) on the projected basis $\pi_2([\tilde{\mathbf{B}}|\mathbf{B}_{[k+1,n]}])$, and so $\pi_{k+2}(\mathbf{B})$ is the same before and after the partial tour. By induction hypothesis, $\pi_2([\tilde{\mathbf{B}}|\mathbf{B}_{[k+1,n]}])$ is k-reduced. □

Lemma 10 gives a terminating condition which ensures that the basis is reduced. We remark that it is even possible to adapt the proof such that it is sufficient to check that

the shape of the projected basis $\pi_{k+1}(\mathbf{B})$ is the same before and after the tour, which is much closer to what one would do in practice to check if *progress was made* (cf. Line 6). However, this requires to relax the definition of SVP-reduction slightly, such that the first vector is not necessarily a shortest vector, but merely a short vector achieving Minkowski's bound. Since this is the only property of SVP reduced bases we need for the analysis below, this does not affect the worst case output quality. Finally, we are aware that it is not obvious that either of these conditions are ever met, e.g. (the shape of) $\pi_{k+1}(\mathbf{B})$ might loop indefinitely. However, in Section 4.1.2 we show that one can put a polynomial upper bound on the number of loops without sacrificing worst case output quality.

To show that the output quality of Self-Dual BKZ in the worst case is at least as good as BKZ's worst case behavior, we analyze the Hermite factor it achieves:

Theorem 4 *If* $\mathbf{B}$ *is k-reduced, then* $\lambda_1(\mathbf{B}_{[1,k]}) \leq \sqrt{\gamma_k}^{\frac{n-1}{k-1}} \cdot \det(\mathbf{B})^{1/n}$.

Proof Assume without loss of generality that $\mathscr{L}(\mathbf{B})$ has determinant 1, and let Δ be the determinant of $\mathscr{L}(\mathbf{B}_{[1,k]})$. Let $\lambda \leq \sqrt{\gamma_k}\Delta^{1/k}$ and $\hat{\lambda} \leq \sqrt{\gamma_k}\Delta^{-1/k}$ be the lengths of the shortest nonzero primal and dual vectors of $\mathscr{L}(\mathbf{B}_{[1,k]})$. We need to prove that $\lambda \leq \sqrt{\gamma_k}^{\frac{n-1}{k-1}}$.

We first show, by induction on n, that the determinant Δ_1 of the first $k-1$ vectors is at most $\sqrt{\gamma_k}^{n-k+1}\det(\mathbf{B})^{(k-1)/n} = \sqrt{\gamma_k}^{n-k+1}$. Since $\mathbf{B}$ is k-reduced, this determinant equals $\Delta_1 = \hat{\lambda} \cdot \Delta \leq \sqrt{\gamma_k}\Delta^{1-1/k}$. (This alone already proves the base case of the induction for $n = k$.) Now, let $\tilde{\mathbf{B}}$ be a SVP reduced basis of $\mathscr{L}(\mathbf{B}_{[1,k]})$ satisfying the k-reduction definition, and consider the determinant $\Delta_2 = \Delta/\lambda$ of $\pi_2(\tilde{\mathbf{B}})$. Since $\pi_2([\tilde{\mathbf{B}}|\mathbf{B}_{[k+1,n]}])$ has determinant $1/\|\tilde{\mathbf{b}}_1\| = 1/\lambda$, by induction hypothesis we have $\Delta_2 \leq \sqrt{\gamma_k}^{n-k}(1/\lambda)^{(k-1)/(n-1)}$.

Multiplying by λ we get

$$\Delta = \lambda\Delta_2 \leq \sqrt{\gamma_k}^{n-k}\lambda^{\frac{n-k}{n-1}} \leq \sqrt{\gamma_k}^{n-k}(\sqrt{\gamma_k}\Delta^{\frac{1}{k}})^{\frac{n-k}{n-1}} = \sqrt{\gamma_k}^{\frac{(n-k)n}{n-1}}\Delta^{\frac{n-k}{k(n-1)}}.$$

Rising both sides to the power $(n-1)/n$ we get $\Delta^{1-\frac{1}{n}} \leq \sqrt{\gamma_n}^{n-k}\Delta^{\frac{1}{k}-\frac{1}{n}}$, or, equivalently, $\Delta^{1-\frac{1}{k}} \leq \sqrt{\gamma_k}^{n-k}$. It follows that $\Delta_1 = \hat{\lambda}\Delta \leq \sqrt{\gamma_k}\Delta^{1-\frac{1}{k}} \leq \sqrt{\gamma_k}^{n-k+1}$, concluding the proof by induction.

We can now prove the main theorem statement. Recall from the inductive proof that $\Delta \leq \sqrt{\gamma_k}^{n-k}\lambda^{\frac{n-k}{n-1}}$. Therefore, $\lambda \leq \sqrt{\gamma_k}\Delta^{1/k} \leq \sqrt{\gamma_k}^{\frac{n}{k}}\lambda^{\frac{n-k}{k(n-1)}}$. Solving for λ, proves the theorem. □

Dynamical System Analysis

Proving a good running time on DBKZ directly seems just as hard as for BKZ, so in this section we analyze the DBKZ algorithm using the dynamical system technique from [32].

Let $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ be an input basis to DBKZ, and assume without loss of generality that $\det(\mathbf{B}) = 1$. During a forward tour, our algorithm computes a sequence of lattice vectors $\mathbf{B}' = [\mathbf{b}'_1, \ldots, \mathbf{b}'_{n-k}]$ where each $\mathbf{b}'_i$ is set to a shortest vector in the projection of $[\mathbf{b}_i, \ldots, \mathbf{b}_{i+k-1}]$ orthogonal to $[\mathbf{b}'_1, \ldots, \mathbf{b}'_{i-1}]$. This set of vectors can be extended to a basis $\mathbf{B}'' = [\mathbf{b}''_1, \ldots, \mathbf{b}''_n]$ for the original lattice. Since $[\mathbf{b}'_1, \ldots, \mathbf{b}'_{i-1}]$ generates a primitive sublattice of $[\mathbf{b}_i, \ldots, \mathbf{b}_{i+k-1}]$, the projected sublattice has determinant $\det(\mathscr{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{i+k-1}))/\det(\mathscr{L}(\mathbf{b}'_1, \ldots, \mathbf{b}'_{i-1}))$, and the length of its shortest vector is

$$\|(\mathbf{b}'_i)^*\| \leq \sqrt{\gamma_k}\left(\frac{\det(\mathscr{L}(\mathbf{b}_1, \ldots, \mathbf{b}_{i+k-1}))}{\det(\mathscr{L}(\mathbf{b}'_1, \ldots, \mathbf{b}'_{i-1}))}\right)^{1/k}. \tag{4.1}$$

At this point, simulations based on the Gaussian Heuristics typically assume that (4.1) holds with equality. In order to get a rigorous analysis without heuristic assumptions, we employ the amortization technique of [33, 32]. For every $i = 1, \ldots, n-k$, let $x_i = \log\det(\mathbf{b}_1, \ldots, \mathbf{b}_{k+i-1})$ and $x'_i = \log\det(\mathbf{b}'_1, \ldots, \mathbf{b}'_i)$. Using (4.1), we get for all

$i = 1, \ldots, n-k$,

$$\begin{aligned} x'_i &= x'_{i-1} + \log \|(\mathbf{b}'_i)^*\| \\ &\leq x'_{i-1} + \alpha + \frac{x_i - x'_{i-1}}{k} \\ &= \omega x'_{i-1} + \alpha + (1-\omega) x_i \end{aligned}$$

where $\omega = (1 - 1/k)$, $\alpha = \frac{1}{2}\log \gamma_k$ and $x'_0 = 0$. By induction on i,

$$x'_i \leq \alpha \frac{1-\omega^i}{1-\omega} + (1-\omega) \sum_{j=1}^{i} \omega^{i-j} x_j,$$

or, in matrix notation $\mathbf{x}' \leq \mathbf{b} + \mathbf{A}\mathbf{x}$ where

$$\mathbf{b} = \alpha k \begin{bmatrix} 1-\omega \\ \vdots \\ 1-\omega^{n-k} \end{bmatrix} \qquad \mathbf{A} = \frac{1}{k} \begin{bmatrix} 1 & & & \\ \omega & 1 & & \\ \vdots & \ddots & \ddots & \\ \omega^{n-k-1} & \cdots & \omega & 1 \end{bmatrix}.$$

Since all the entries of $\mathbf{A}$ are positive, we also see that if $X_i \geq x_i$ are upper bounds on the initial values x_i for all i, then the vector $X' = \mathbf{A}X + \mathbf{b}$ gives upper bounds on the output values $x'_i \leq X'_i$.

The vector $\mathbf{x}'$ describes the shape of the basis matrix before the execution of a backward tour. Using lattice duality, the backward tour can be equivalently formulated by the following steps:

1. Compute the reversed dual basis $\mathbf{D}$ of $\mathbf{B}'$
2. Apply a forward tour to $\mathbf{D}$ to obtain a new dual basis $\mathbf{D}'$
3. Compute the reversed dual basis of $\mathbf{D}'$

The reversed dual basis computation yields a basis $\mathbf{D}$ such that, for all $i = 1, \ldots, n-k$,

$$\begin{aligned} y_i &= \log\det(\mathbf{d}_1, \ldots, \mathbf{d}_{k+i-1}) \\ &= -\log(\det(\mathbf{B}')/\det([\mathbf{b}'_1, \ldots, \mathbf{b}'_{n-k+1-i}])) \\ &= \log\det([\mathbf{b}'_1, \ldots, \mathbf{b}'_{n-k+1-i}]) = x'_{n-k+1-i}. \end{aligned}$$

So, the vector $\mathbf{y}$ describing the shape of the dual basis at the beginning of the backward tour is just the reverse of $\mathbf{x}'$. It follows that applying a full (forward and backward) DBKZ tour produces a basis such that if X are upper bounds on the log determinants $\mathbf{x}$ of the input matrix, then the log determinants of the output matrix are bounded from above by

$$\mathbf{R}(\mathbf{A}\mathbf{R}(\mathbf{A}X + \mathbf{b}) + \mathbf{b}) = (\mathbf{R}\mathbf{A})^2 X + (\mathbf{R}\mathbf{A} + \mathbf{I})\mathbf{R}\mathbf{b}$$

where $\mathbf{R}$ is the coordinate reversal permutation matrix. This leads to the study of the discrete time affine dynamical system

$$X \mapsto (\mathbf{R}\mathbf{A})^2 X + (\mathbf{R}\mathbf{A} + \mathbf{I})\mathbf{R}\mathbf{b}. \tag{4.2}$$

Output Quality

We first prove that this system has at most one fixed point.

Claim 1 *The dynamical system (4.2) has at most one fixed point.*

Proof Any fixed point is a solution to the linear system $((\mathbf{R}\mathbf{A})^2 - \mathbf{I})X + (\mathbf{R}\mathbf{A} + \mathbf{I})\mathbf{R}\mathbf{b} = \mathbf{0}$. To prove uniqueness, we show that the matrix $((\mathbf{R}\mathbf{A})^2 - \mathbf{I})$ is non-singular, i.e., if $(\mathbf{R}\mathbf{A})^2\mathbf{x} = \mathbf{x}$ then $\mathbf{x} = \mathbf{0}$. Notice that the matrix $\mathbf{R}\mathbf{A}$ is symmetric, so we have $(\mathbf{R}\mathbf{A})^2 = (\mathbf{R}\mathbf{A})^T\mathbf{R}\mathbf{A} = \mathbf{A}^T\mathbf{A}$. So proving $((\mathbf{R}\mathbf{A})^2 - \mathbf{I})$ is non-singular is equivalent to showing that 1 is not an eigenvalue of $\mathbf{A}^T\mathbf{A}$. We have $\rho(\mathbf{A}^T\mathbf{A}) = \|\mathbf{A}\|_2^2 \leq \|\mathbf{A}\|_1\|\mathbf{A}\|_\infty$, where $\rho(\cdot)$

denotes the spectral radius of the given matrix (i.e. the largest eigenvalue in absolute value). But we also have

$$\|\mathbf{A}\|_\infty = \|\mathbf{A}\|_1 = \frac{1}{k}\sum_{i=0}^{n-k-1}\omega^i = \frac{1-\omega^{n-k}}{k(1-\omega)} = 1-\omega^{n-k} < 1 \tag{4.3}$$

which shows that the absolute value of any eigenvalue of $\mathbf{A}^T\mathbf{A}$ is strictly smaller than 1.□

In order to analyze the output quality of DBKZ, we need to find a fixed point for (4.2). We proved that $(\mathbf{RA})^2 - \mathbf{I}$ is a non-singular matrix. Since $(\mathbf{RA})^2 - \mathbf{I} = (\mathbf{RA}+\mathbf{I})(\mathbf{RA}-\mathbf{I})$, it follows that $(\mathbf{RA}\pm\mathbf{I})$ are also non-singular. So, we can factor $(\mathbf{RA}+\mathbf{I})$ out of the fixed point equation $((\mathbf{RA})^2-\mathbf{I})\mathbf{x}+(\mathbf{RA}+\mathbf{I})\mathbf{Rb} = \mathbf{0}$, and obtain $(\mathbf{RA}-\mathbf{I})\mathbf{x}+\mathbf{Rb} = 0$. This shows that the only fixed point of the full dynamical system (if it exists) must also be a fixed point of a forward tour $\mathbf{x} \mapsto \mathbf{R}(\mathbf{Ax}+\mathbf{b})$.

Claim 2 *The fixed point of the dynamical system* $\mathbf{x} \mapsto \mathbf{R}(\mathbf{Ax}+\mathbf{b})$ *is given by*

$$x_i = \frac{(n-k-i+1)(k+i-1)}{k-1}\alpha. \tag{4.4}$$

Proof The unique fixed point of the system is given by the solution to the linear system $(\mathbf{R}-\mathbf{A})\mathbf{x} = \mathbf{b}$. We prove that (4.4) is a solution to the system by induction on the rows. For the first row, the system yields

$$x_{n-k} - x_1/k = \alpha. \tag{4.5}$$

From (4.4) we get that $x_{n-k} = \frac{n-1}{k-1}\alpha$ and $x_1 = \frac{k(n-k)}{k-1}\alpha$. Substituting these into (4.5), the validity is easily verified.

The r-th row of the system is given by

$$x_{n-k-r+1} - \frac{1}{k}\left(\sum_{j=1}^{r} \omega^{r-j} x_j\right) = \frac{1-\omega^r}{1-\omega}\alpha \tag{4.6}$$

which is equivalent to

$$x_{n-k-r+1} + \omega\left(x_{n-k-r+2} - \frac{1}{k}\left(\sum_{j=1}^{r-1} \omega^{r-1-j} x_j\right)\right) - \frac{x_r}{k} - \omega x_{n-k-r+2} = \frac{1-\omega^r}{1-\omega}\alpha. \tag{4.7}$$

By induction hypothesis, this is equivalent to

$$\omega\left(\frac{1-\omega^{r-1}}{1-\omega}\right)\alpha + x_{n-k-r+1} - \frac{x_r}{k} - \omega x_{n-k-r+2} = \frac{1-\omega^r}{1-\omega}\alpha. \tag{4.8}$$

Substituting (4.4) in for $i = n-k-r+1$, r, and $n-k-r+2$, we get

$$x_{n-k-r+1} - \frac{x_r}{k} - \omega x_{n-k-r+2} = \frac{kr(n-r) - (n-r-k+1)(r+k-1) - (k-1)(r-1)(n-r+1)}{k(k-1)}\alpha$$

which, after some tedious, but straight forward, calculation can be shown to be equal to α (i.e. the fraction simplifies to 1). This in turn shows that the left hand side of (4.8) is equivalent to

$$\omega\left(\frac{1-\omega^{r-1}}{1-\omega}\right)\alpha + \alpha$$

which is equal to its right hand side. □

Note that since x_1 corresponds to the log determinant of the first block, applying Minkowski's theorem results in the same worst case Hermite factor as proved in Theorem 4.

Convergence

We now consider the convergence rate of the system, which will allow us to deduce statements about the running time of DBKZ. Consider any input vector $\mathbf{v}$ and write it as $\mathbf{v} = \mathbf{x} + \mathbf{e}$, where $\mathbf{x}$ is the fixed point of the dynamical system as in (4.4). The system sends $\mathbf{v}$ to $\mathbf{v} \mapsto \mathbf{RAv} + \mathbf{b} = \mathbf{RAx} + \mathbf{RAe} + \mathbf{b} = \mathbf{x} + \mathbf{RAe}$, so the difference $\mathbf{e}$ to the fixed point is mapped to $\mathbf{RAe}$ in each iteration. In order to analyze the convergence of the algorithm, we consider the induced norm of the matrix $\|\mathbf{RA}\|_p = \|\mathbf{A}\|_p$, since after t iterations the difference is $(\mathbf{RA})^t\mathbf{e}$ and so its norm is bounded by $\|(\mathbf{RA})^t\mathbf{e}\|_p \leq \|(\mathbf{RA})^t\|_p\|\mathbf{e}\|_p \leq \|\mathbf{RA}\|_p^t\|\mathbf{e}\|_p$. So if the induced norm of $\mathbf{A}$ is strictly smaller than 1, the corresponding norm of the error vector follows an exponential decay. While the spectral norm of $\mathbf{A}$ seems hard to bound, the 1 and the infinity norm are straight forward to analyze. In particular, we saw in (4.3) that $\|\mathbf{A}\|_\infty = 1 - \omega^{n-k}$. This proves that the algorithm converges. Furthermore, let the input be a basis $\mathbf{B}$ (with $\det(\mathbf{B}) = 1$), the corresponding vector $\mathbf{v} = (\log\det(\mathbf{b}_1, \ldots, \mathbf{b}_{k+i-1}))_{1\leq i\leq n}$ and write $\mathbf{v} = \mathbf{x} + \mathbf{e}$. Then we have $\|\mathbf{e}\|_\infty = \|\mathbf{v} - \mathbf{x}\|_\infty \leq \|\mathbf{v}\|_\infty + \|\mathbf{x}\|_\infty \leq \mathrm{poly}(n, \mathrm{size}(\mathbf{B}))$. This implies that for

$$t = \mathrm{polylog}(n, \mathrm{size}(\mathbf{B}))/\omega^{n-k} \approx O(e^{(n-k)/k})\mathrm{polylog}(n, \mathrm{size}(\mathbf{B})) \tag{4.9}$$

we have that $\|(\mathbf{RA})^t\mathbf{e}\| \leq c$ for constant c. Equation (4.9) already shows that for $k = \Omega(n)$, the algorithm converges in a number of tours polylogarithmic in the lattice dimension n, i.e. makes at most $\tilde{O}(n)$ SVP calls.

In the initial version of [54], proving polynomial convergence for arbitrary k was left as an open problem. Neumaier filled this gap in [58]. In what follows we reformulate his proof using our notation for completeness, but we stress that this is originally Neumaier's work.

Claim 3 *Let* $\mathbf{x}$ *be the fixed point of the dynamical system as defined in Claim 2 and*

$r_i = e_i/x_i$ be the relative error of the dynamical system. Then, if $\mathbf{r}, \mathbf{r}'$ are the relative errors before and after the execution of one iteration of the system, then $\|\mathbf{r}'\|_\infty \le (1-\varepsilon)\|\mathbf{r}\|_\infty$ for $\varepsilon = 1/(1+n^2/(4k(k-1))) \approx (2k/n)^2$. When $k \ge n/2$, it is enough to take $\varepsilon = (k-1)/(n-1)$.

Proof Assume without loss of generality that $\|\mathbf{r}\|_\infty = (k-1)/\alpha$, i.e., $|e_i| \le ((k-1)/\alpha)x_i = (n-k-i+1)(k+i-1)$ for all $i = 1,\ldots,n-k$. We need to prove that $|r'_j| = |(\mathbf{RAe})_j|/x_j \le ((k-1)/\alpha)(1-\varepsilon)$ for all j, or, equivalently,

$$|(\mathbf{Ae})_j| = |(\mathbf{RAe})_{n-k-j+1}| \le \frac{k-1}{\alpha}(1-\varepsilon)x_{n-k-j+1} = j(n-j)(1-\varepsilon).$$

By the definition of $\mathbf{A}$, we have

$$|(\mathbf{Ae})_j| \le \frac{1}{k}\sum_{i=1}^{j} \omega^{j-i}|e_i| \le \frac{1}{k}\sum_{i=1}^{j} \omega^{j-i}(n-k-i+1)(k+i-1) \equiv f(j).$$

So, it is enough to prove that the function $f(j)$ on the right hand side satisfies $f(j) \le j(n-j)(1-\varepsilon)$. We prove this inequality by induction on j, under the assumption that

$$\varepsilon \le g(j) \equiv \frac{k(k-1)}{k(n-2j-1)+j(n-j)}.$$

- base case ($j = 1$): $f(1) = n-k \le (n-1)(1-\varepsilon)$ if and only if $\varepsilon \le g(0) = (k-1)/(n-1)$.
- inductive step: assume $f(j) \le j(n-j)(1-\varepsilon)$ by inductive hypothesis. We need to prove that

$$\begin{aligned} f(j+1) &= \omega f(j) + \frac{(n-k-j)(k+j)}{k} \\ &\le \frac{k-1}{k} j(n-j)(1-\varepsilon) + \frac{(n-k-j)(k+j)}{k} \end{aligned}$$

is at most $(j+1)(n-(j+1))(1-\varepsilon)$. This is true if and only if $\varepsilon \leq g(j)$.

This concludes the inductive proof, as long as $\varepsilon \leq g(j)$ for $j = 0, \ldots, (n-k)$. Finally, we observe that the function $g(j)$ is minimized at $j = \frac{n}{2} - k$, with minimum $g(\frac{n}{2} - k) = 1/(1 + n^2/(4k(k-1)))$. When $k > n/2$, the minimum is achieved at $j < 0$ and $g(j)$ is monotonically increasing for $j \leq 0$. So it is enough to take $\varepsilon = g(0) = (k-1)/(n-1)$.□

By a similar argument as above, this shows that the error can be made arbitrarily close to 0 in $O((n/k)^2)\text{polylog}(n, \text{size}(\mathbf{B}))$ tours.

Heuristic Analysis

In the context of cryptanalysis, we are more interested in the average-case behavior of algorithms. For this we can use a very simple observation to predict the Hermite factor achieved by DBKZ. Note that the proof of Theorem 4 is based solely on Minkowski's bound $\lambda_1(\mathbf{B}) \leq \sqrt{\gamma_n}\det(\mathbf{B})^{1/n}$. Replacing it with Heuristic 1 yields the following corollary.

Corollary 3 *Applying Heuristic 1 to every lattice that is passed to the SVP oracle during the execution of Algorithm 4.1, if* $\mathbf{B}$ *is k-reduced, then* $\lambda_1(\mathbf{B}_{1,k}) = GH(k)^{\frac{n-1}{k-1}}\det(\mathbf{B})^{1/n}$.

As the Hermite factor is the most relevant quantity in many cryptanalytic settings, Corollary 3 is already sufficient for many intended applications in terms of output quality. We remark that the proof of achieved worst-case output quality of Slide reduction also only relies on Minkowski's bound. This means the same observation can be used to predict the average case behavior of Slide reduction and yields the same estimate as Corollary 3. In fact, from the recursive definition of Slide reduction it is clear that this yields even more information about the returned basis: we can use Corollary 3 to predict the norm of $\|\mathbf{b}_{ik+1}\|$ for all $i \in [n/k]$. A short calculation shows that these vectors follow

a geometric series, supporting a frequently assumed behavior of lattice reduction, namely the *Geometric Series Assumption* [73].

However, many attacks [44, 62] require to estimate the average-case output much more precisely. Fortunately, applying a similar trick as in Corollary 3 to the dynamical systems analysis in Section 4.1.2 allows us to obtain much more information about the basis. For this, note that again we can replace Minkowski's theorem in the analysis by Heuristic 1. This transformation changes the dynamical system in (4.2) only slightly, the only difference being that $\alpha = \frac{1}{2}\log GH(k)$. As the analysis is independent of the constant α, we can translate the fixed point in (4.4) to information about the shape of the basis that DBKZ is likely to return.

Corollary 4 *Applying Heuristic 1 to every lattice that is passed to the SVP oracle during the execution of Algorithm 4.1, the fixed point of the heuristic dynamical system, i.e. (4.2) with $\alpha = \frac{1}{2}\log GH(k)$, is (4.4) with the same α and implies that after one more forward tour, the basis satisfies*

$$\|\mathbf{b}_i^*\| = GH(k)^{\frac{n+1-2i}{2(k-1)}} \det(\mathscr{L}(\mathbf{B}))^{\frac{1}{n}} \tag{4.10}$$

for all $i \leq n-k$.

Proof According to (4.4), upon termination of Algorithm 4.1 the output basis satisfies

$$\log(\det([\mathbf{b}_1,\ldots,\mathbf{b}_i])) = \frac{(n-k-i+1)(k+i-1)}{k-1}\alpha$$

By Heuristic 1 we have $\log\|\mathbf{b}_1\| = \alpha + x_1/k$, from which Equation (4.10) easily follows for $i = 1$. Now assume (4.10) holds for all $j < i$. Then we have, again by Heuristic 1, $\log\|\mathbf{b}_i^*\| = \alpha + (x_i - \sum_{j<i}\log\|\mathbf{b}_j^*\|)/k$. Invoking the induction hypothesis, Equation (4.10) easily follows for all $i \leq n-k$. □

Corollary 4 shows that the output of the DBKZ algorithm, if terminated after a forward tour, can be expected to closely follow the GSA, at least for all $i \leq n-k$ and can be computed using simple closed formulas. It is noteworthy that the self-dual properties of DBKZ imply that if terminated after a backward tour, the GSA holds for all $i \geq k$. This means, depending on the application one can choose which part of the output basis to predict. Moreover, we see that DBKZ allows to predict a much larger part of the basis than Slide reduction solely based on the Gaussian Heuristic. If one is willing to make additional assumptions, i.e. assumptions about the shape of a k-dimensional HKZ reduced basis, the BKZ simulator allows to predict the shape of the entire basis output by BKZ. Obviously, the same assumptions can be used to estimate the remaining parts of the shape of the basis in the case of Slide reduction and DBKZ, since a final application of a HKZ reduction to individual blocks of size k only requires negligible amount of time compared to the running time of the entire algorithm. Furthermore, since the estimation of the known part of the shape (from Corollary 3 and 4) do not depend on these additional assumptions, the estimation for Slide reduction and DBKZ is much less sensitive to the (in-)correctness of these assumptions, while errors propagate during the BKZ simulation.

To compare the expected output of BKZ, DBKZ, and Slide reduction, we generated a Goldstein-Mayer lattice [30] in dimension $n = 200$ with numbers of bit size 2000, applied LLL to it, and simulated the execution of BKZ with block size $k = 100$ until no more progress was made. The output in terms of the logarithm of the shape of the basis for the first 100 basis vectors is shown in Figure 4.1 and compared to the GSA. Recall that the latter represents the expected output of DBKZ and, to some degree, Slide reduction. Under the assumption that Heuristic 1 and the BKZ simulator are accurate, one would expect BKZ to behave a little worse than the other two algorithms in terms of output quality.

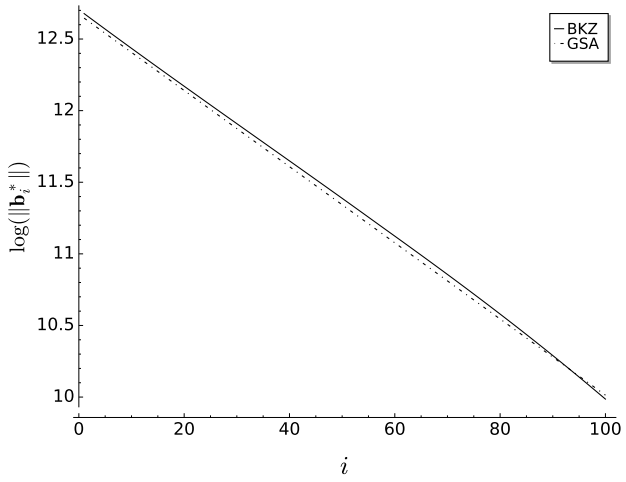


Figure 4.1. Expected shape of the first 100 basis vectors in dimension $n = 200$ after BKZ compared to the GSA. Note that the latter corresponds exactly to the expected shape of the first 100 basis vectors after DBKZ (cf. Corollary 4).

Experiments

For an experimental comparison, we implemented DBKZ and Slide reduction in `fpLLL`. At the point of this writing, the implementation of both algorithms was contributed to `fpLLL` and is now readily available in stable versions. SVP reduction in `fpLLL` is implemented in the standard way as described in Section 2.4.1. For dual SVP reduction we used the algorithm explained in the Section 3.1.

Methodology

In the context of cryptanalysis we are usually interested in the root Hermite factor achievable using lattice reduction in order to choose parameters for cryptosystems, as this often determines the success probability and/or complexity of an attack. It is clear that merely reporting on the average root Hermite factor achieved is of limited use for this. Instead we will view the resulting root Hermite factor achieved by a certain reduction algorithm (with certain parameters) as a random variable and try to estimate the main statistical parameters of its distribution. We believe this will eventually allow

for more meaningful security estimates. The only previous experimental work studying properties of the underlying distribution of the root Hermite factor [24] suggests that it is Gaussian-like but the study is limited to relatively small block sizes.

Since experiments with lattice reduction are rather time consuming, it is infeasible to generate as much data as desirable to estimate statistical parameters like the mean value and standard deviation accurately. A standard statistical technique to overcome this is to use bootstrapping to compute confidence intervals for these parameters. Roughly speaking, in order to compute the confidence interval for an estimator from a set of N samples, we sample l sets of size N with replacement from the original samples and compute the estimator for each of them. Intuitively, this should give a sense of the variability of the estimator computed on the samples. Our confidence interval with confidence parameter α, according to the bootstrap percentile interval method, is simply the $\alpha/2$ and $1-\alpha/2$ quantiles. For further discussion we refer to [80]. Throughout this work we use $\alpha = .05$ and $l = 100$. The complete confidence intervals for mean value and standard deviation can be found in [54, 53]. Whenever we refer to the standard deviation of a distribution resulting from the application of a reduction algorithm and computing the root Hermite factor achieved, we mean the maximum of the corresponding confidence interval.

It is folklore that the output quality of lattice reduction algorithms measured in the root Hermite factor depends mostly on the block size parameter rather than on properties of the input lattice, like the dimension or bit size of the numbers, at least when the lattice dimension and size of the numbers is large enough. A natural approach to comparing the different algorithms would be to fix a number of lattices of certain dimension and bit size and run the different algorithms with varying block size on them. Unfortunately, Slide reduction requires the block size to divide the dimension.[1] To circumvent this we

[1]While it is trivial to generalize Slide reduction to other block sizes, the performance in terms of

select the dimension of the input lattices depending on the block sizes we want to test, i.e. $n = t \cdot k$, where k is the block size and t is a small integer. This is justified as most lattice attacks involve choosing a suitable sublattice to attack, where such a requirement can easily be taken into account. Since for very small dimensions block reduction performs a little better then in larger dimensions, we need to deal with a trade-off here: on the one hand we need to ensure that the lattice dimension n is large enough, even for small block sizes, so that the result is not biased positively for small block sizes due to the small dimension. On the other hand, if the lattice dimension grows very large we would have to increase the precision of the GSO computation significantly which would result in an artificial slow down and thus limit the data we are able to collect. Our experiments and previous work [24] suggest that the bias for small dimensions weakens sufficiently as soon as the lattice dimension is larger than 140, so for the lattice dimension n we select the smallest multiple t of the block size k such that $t \cdot k \geq 140$.

For each block size we generated 10 different subset sum lattices in dimension n in the sense of [33] and we fix the bit size of the numbers to $10 \cdot n$ following previous work [33, 52]. Experimental studies [61] have shown that this notion of random lattices is suitable in this context as lattice reduction behaves similarly on them as on "random" lattices in a mathematically more precise sense [30][2]. Then we ran each of the three reduction algorithms with corresponding block size on each of those lattices. For BKZ and DBKZ we used the same terminating condition: the algorithms terminate when the slope of the shape of the basis does not improve during 5 loop iterations in a row (this is the default terminating condition in `fpLLL`'s BKZ routine with auto abort option set). Finally, for sufficiently large block sizes ($k > 45$), we preprocessed the local blocks with BKZ-$(k/2)$ before calling the SVP oracle, since this has been shown to achieve good

the achieved output quality of the basis deteriorates somewhat in this case compared to other reduction algorithms [43].

[2] In fact, subset sum lattices are extremely similar to the random lattices of [30].

asymptotic running time (cf. Section 3.3.2) and also seemed a good choice in practice in our experiments.

Results

Figure 4.2 shows the average output quality including the confidence interval produced by each of the three algorithms in comparison with the prediction based on the Gaussian Heuristic (cf. Equation (2.7)). It demonstrates that BKZ and DBKZ have comparable performance in terms of output quality and clearly outperform Slide reduction for small block sizes (< 50), which confirms previous reports [24]. For some of the small block sizes (e.g. $k = 35$) BKZ seems to perform unexpectedly well in our experiments. To see if this is indeed inherent to the algorithms or a statistical outlier owed to the relatively small number of data points, we ran some more experiments with small block sizes. We report on the results in [53], where we show that the performance of BKZ and DBKZ are actually extremely close for these parameters.

Furthermore, Figure 4.2 shows that all three algorithms tend towards the prediction given by Equation (2.7) in larger block sizes, supporting the conjecture, and Slide reduction becomes quite competitive. Even though BKZ still seems to have a slight edge for block size 75, note that the confidence intervals for Slide reduction and BKZ are heavily overlapping here. This is in contrast to the only previous study that involved Slide reduction [24], where Slide reduction was reported to be entirely noncompetitive in practice and thus mainly of theoretical interest.

Figure 4.3 shows the same data separately for each of the three algorithms including estimated standard deviation. The data does not seem to suggest that one or the other algorithm behaves "nicer" with respect to predictability – the standard deviation ranges between 0.0002 and 0.0004 for all algorithms, but can be as high as 0.00054. Note that while these numbers might seem small, it affects the base of the exponential that the

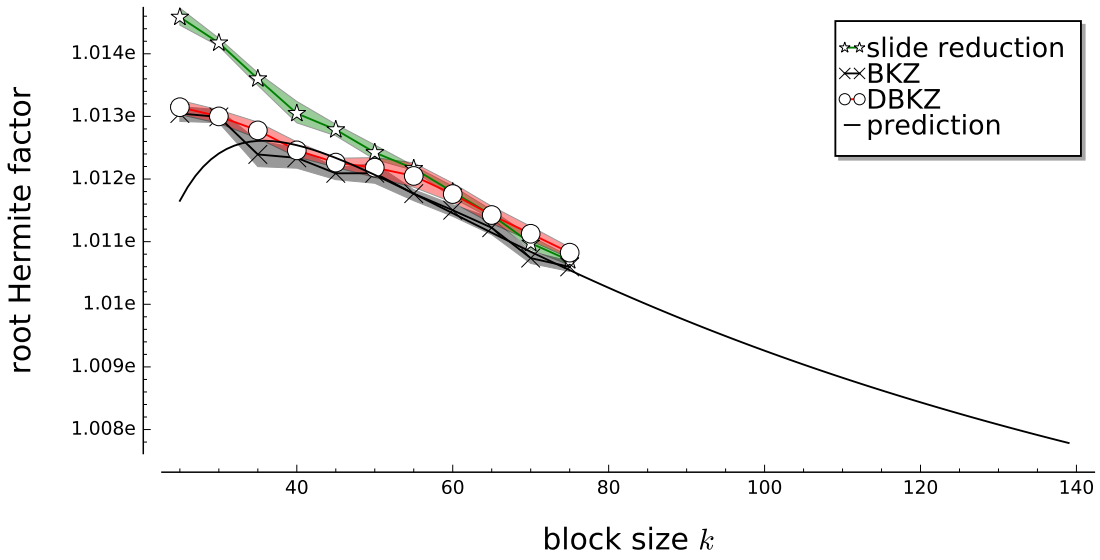
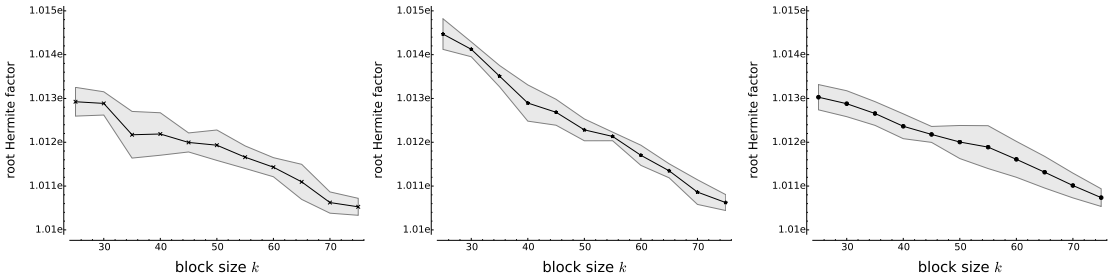


Figure 4.2. Confidence interval of average root Hermite factor for random bases as computed by different reduction algorithms and the prediction given by Equation (2.7).

short vector is measured in, so small changes have a large impact. The standard deviation varies across different block sizes, but there is no evidence that it might converge to smaller values or even 0 in larger block sizes. So we have to assume, that it remains a significant factor for larger block sizes and should be taken into account in cryptanalysis. It is entirely conceivable that the application of a reduction algorithm yields a root Hermite factor significantly smaller than the corresponding mean value.

In order to compare the runtime of the algorithms we ran separate experiments,



(a) BKZ **(b)** Slide Reduction **(c)** DBKZ

Figure 4.3. Same as Figure 4.2 with estimated standard deviation

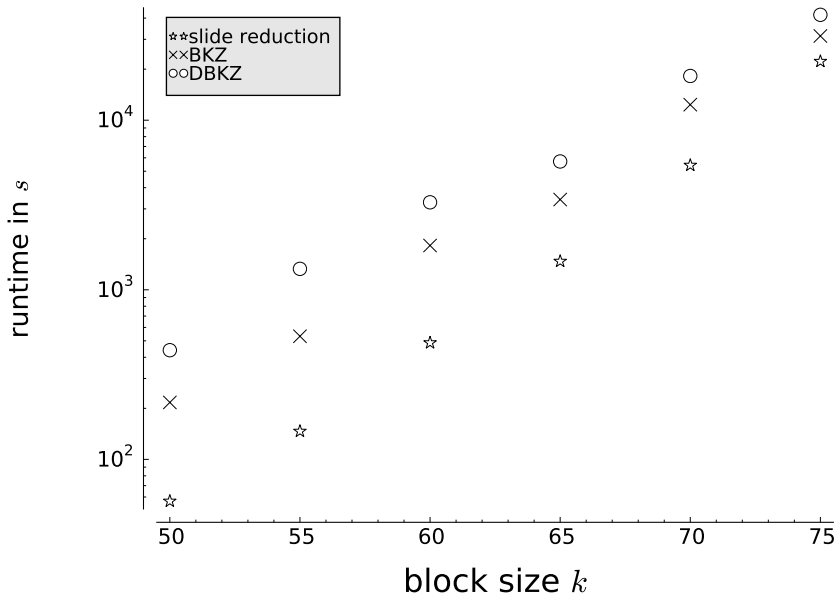


Figure 4.4. Average runtime in seconds for random bases in dimension $n = 2k$ for different reduction algorithms (in log scale).

because due to the way we selected the dimension, the data would exhibit a somewhat strange "zigzag" behavior. For each block size $50 \leq k \leq 75$ we generated again 10 random subset sum lattices with dimension $n = 2k$ and the bit size of the numbers was fixed to 1400. Figure 4.4 shows the average runtime for each of the algorithms and block size in log scale. It shows that the runtime of all three algorithms follows a close to single exponential (in the block size) curve. This supports the intuition that the runtime mainly depends on the complexity of the SVP oracle, since we are using an implementation that preprocesses the local blocks before enumeration with large block size. As we showed in Section 3.3.2 this achieves an almost single exponential complexity (up to logarithmic factors in the exponent).

The data also shows that in terms of runtime, Slide reduction outperforms both, BKZ and DBKZ. But again, with increasing block size the runtime of the different algorithms seem to converge to each other. Combined with the data from Figure 4.2 this suggests that all three algorithms offer a similar trade-off between runtime and

achieved Hermite factor for large block sizes. This shows that Slide reduction is not only theoretically interesting with its cleaner and tighter analysis of both, output quality and runtime, but also quite competitive in practice. It should be noted that we analyzed Slide reduction as described in [23]. While significant research effort has been spent on improving BKZ, essentially nothing along these lines has been done with regards to Slide reduction. We hope that the results reported here will initiate more research into improvements, both in practice and theory, of Slide reduction.

Chapter 4, in full, is a reprint of material as it appears (with minor modifications) in the paper "Practical, Predictable Lattice Basis Reduction" [54] by Daniele Micciancio and Michael Walter, published in the proceedings of the Thirty-Fifth Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2016). The dissertation author was the primary investigator and author of this paper.

Part II

Secure Discrete Gaussian Sampling

Chapter 5

The Bit Security of Cryptographic Primitives

In this chapter we define and analyze a notion of bit security, which we will use in the following chapters to reason about the security of cryptographic schemes.

Security Games

In this section we formally define the bit security of cryptographic primitives in a way that captures practical intuition and is theoretically sound. As the security of cryptographic primitives is commonly defined using games, we start by defining a general class of security games.

Definition 24 *An n-bit* security game *is played by an adversary A interacting with a challenger X. At the beginning of the game, the challenger chooses a secret x, represented by the random variable $X \in \{0,1\}^n$, from some distribution $\mathscr{D}_X$. At the end of the game, A outputs some value, which is represented by the random variable A. The goal of the adversary is to output a value a such that $R(x,a)$, where R is some relation. A may output a special symbol $\perp$ such that $R(x,\perp)$ and $\bar{R}(x,\perp)$ are both false.*

This definition is very general and covers a lot of standard games from the literature. Some illustrative examples are given in Table 5.1. But for the cryptographic

primitives explicitly studied in this work, it will be enough to consider the simplest version of the definition where $R = \{(x,x)|x \in X\}$ is the identity relation, i.e., the goal of the adversary is to guess the secret x. We formally define the indistinguishability game for two distributions because we refer to it extensively throughout this work.

Definition 25 *Let* $\{\mathscr{D}_0^\theta\}_\theta$, $\{\mathscr{D}_1^\theta\}_\theta$ *be two distribution ensembles. The* indistinguishability game *is defined as follows: the challenger C chooses* $b \leftarrow \mathscr{U}(\{0,1\})$. *At any time after that the adversary A may (adaptively) request samples by sending* θ_i *to C, upon which C draws samples* $c_i \leftarrow \mathscr{D}_b^{\theta_i}$ *and sends* c_i *to A. The goal of the adversary is to output* $b' = b$.

We loosely classify primitives into two categories according to their associated security games: we call primitives, where the associated security game is a 1-bit game ($O(\kappa)$-bit game), *decision primitives* (*search primitive*, respectively).

Note that we allow the adversary to always output $\perp$, which roughly means "I don't know", even for decision primitives. This is a crucial difference from previous definitions that force the distinguisher to always output a bit. The reason this is important is that in games, where the distinguisher is not able to check if it produced the correct result, it is more informative to admit defeat rather than guessing at random. In many cases this will allow for much tighter reductions (cf. Section 5.3.2). Such a definition in the context of indistinguishability games is not entirely new, as Goldreich and Levin [29, 42] also allowed this type of flexibility for the distinguisher. To the best of our knowledge, this is the only place this has previously appeared in the cryptographic literature.

Before defining the advantage of an adversary, we specify a few properties of its output distribution.

Definition 26 *For any adversary A playing a security game, we define its* output proba-

Table 5.1. Typical instantiations of security games covered by Definition 24. The security parameter is denoted by κ. In the definition of digital signatures, the list Q of the adversary's queries are regarded as part of its output.

Game	R	n	$\mathscr{D}_X$
Uninvertibility of one-way permutations	$\{(x,y) \mid x = y\}$	$O(\kappa)$	$\mathscr{U}$
Uninvertibility of one-way functions f	$\{(x,y) \mid f(x) = f(y)\}$	$O(\kappa)$	$\mathscr{U}$
Pre-image resistance for hash functions h	$\{(x,y) \mid x \neq y, h(x) = h(y)\}$	$O(\kappa)$	$\mathscr{U}$
Indistinguishability of two distributions	$\{(x,y) \mid x = y\}$	1	$\mathscr{U}$
Unforgeability of signature scheme (K,S,V)	$\{(x,(m,\sigma,Q)) \mid (pk,sk) \leftarrow K(x), V(pk,m,\sigma) = 1, m \notin Q\}$	$O(\kappa)$	$K(\mathscr{U})$

bility *as* $\alpha^A = \Pr[A \neq \perp]$, *its* conditional success probability *as* $\beta^A = \Pr[R(X,A)|A \neq \perp]$, *where the probabilities are taken over the randomness of the entire security game (including the internal randomness of A). Finally, in the context of decision primitives, we also define A's* conditional distinguishing advantage *as* $\delta^A = 2\beta^A - 1$. *With all of these quantities, when the adversary A is clear from context, we drop the corresponding superscript.*

Now we are ready to define the advantage. The definition is trying to capture the amount of information that the adversary is able to learn about the secret. For this we use tools from information theory to quantify exactly this information. A straightforward definition could try to measure the mutual information between the random variables X (modeling the secret) and A (modeling the adversary output, cf. Definition 24). Unfortunately, the variable A might reveal X completely in an information theoretical sense, yet not anything in a computational sense. To break any computationally hidden connection between X and A, we introduce another random variable Y, which indicates, when A actually achieves its goal and otherwise does not reveal anything about the secret.

Definition 27 *For any security game with corresponding random variable X and A, the*

adversary's advantage *is*

$$\mathrm{adv}^A = \frac{I(X;Y)}{H(X)} = 1 - \frac{H(X|Y)}{H(X)}$$

where $I(\cdot;\cdot)$ is the mutual information, $H(\cdot)$ is the Shannon entropy, and $Y(X,A)$ is the random variable with marginal distributions $Y_{x,a} = \{Y \mid X = x, A = a\}$ defined as

1. $Y_{x,\perp} = \perp$, *for all* x.
2. $Y_{x,a} = x$, *for all* $(x,a) \in R$.
3. $Y_{x,a} = \{x' \leftarrow \mathscr{D}_X \mid x' \neq x\}$, *for all* $(x,a) \in \bar{R}$.

At first glance, the definition of Y might not be obviously intuitive, except for case 1. For case 2, note that x completely determines the set $R(x,\cdot)$ and if the adversary finds an element in it, then it wins the game. Therefore, one can think of $R(x,\cdot)$ as a secret set, and finding any element in it as completely breaking the scheme. Finally, the third case defines Y to follow the distribution of the secret, but is conditioned on the event that it is incorrect. The intuition here is that if an adversary outputs something, then his goal is to bias the secret distribution towards the correct one, i.e. it will allow us to quantify how much better A performs than random guessing.

With the definition of the advantage in place, the definition of bit security follows quite naturally.

Definition 28 *Let $T : \{A \mid A \text{ is any algorithm}\} \mapsto \mathbb{R}$ be a measure of resources that is linear under repetition, i.e. $T(kA) = kT(A)$, where kA is the k time repetition of A. For any primitive, we define its* bit security *as* $\min_A \log \frac{T(A)}{adv^A}$.

For convenience we will often write $T(A)$ as T^A or simply T if A is clear from context. Note that we leave out a concrete definition of the resources on purpose, since

we focus on the advantage. Our definition can be used with many different measures, for example running time, space, advice, etc., or combinations of them.

The Adversary's Advantage

While the advantage as defined in the previous section captures the intuition about how well an adversary performs, it seems too complex to be handled in actual proofs or to be used in practice. A simple definition in terms of simple quantities related to the adversary (e.g. α^A and β^A) would be much more desirable. The goal of this section is to distill such a definition by considering a broad and natural class of adversaries and games.

Theorem 5 *For any n-bit security game with uniform secret distribution, let A be an adversary that for any secret $x \in \{0,1\}^n$ outputs $\perp$ with probability $1-\alpha$, some value a such that $R(x,a)$ with probability $\beta\alpha$, and some value a such that $\bar{R}(x,a)$ with probability $(1-\beta)\alpha$. Then*

$$\mathrm{adv}^A = \alpha\left(1 - \frac{(1-\beta)\log(2^n-1) + H(\mathscr{B}_\beta)}{n}\right). \tag{5.1}$$

Proof From the definition of Y in Definition 27 we get for any $x, y \in \{0,1\}^n$ with $y \neq x$

$$\Pr[Y = \perp | X = x] = 1 - \alpha$$
$$\Pr[Y = x | X = x] = \alpha\beta$$
$$\Pr[Y = y | X = x] = \frac{\alpha(1-\beta)}{2^n - 1}$$

From this we compute

$$\Pr[Y=\perp] = 1-\alpha$$

$$\begin{aligned}\Pr[Y=y] &= \Pr[Y=y|X=y]\Pr[X=y] + \Pr[Y=y|X\neq y]\Pr[X\neq y] \\ &= \frac{\alpha\beta}{2^n} + \frac{2^n-1}{2^n}\frac{\alpha(1-\beta)}{2^n-1} \\ &= \frac{\alpha}{2^n}\end{aligned}$$

Now we calculate the conditional entropy

$$\begin{aligned}H(X|Y) =& \sum_{x,y}\Pr[Y=y|X=x]\Pr[X=x]\log\frac{\Pr[Y=y]}{\Pr[Y=y|X=x]\Pr[X=x]} \\ =& \sum_{x}\Pr[Y=\perp|X=x]\Pr[X=x]\log\frac{\Pr[Y=\perp]}{\Pr[Y=\perp|X=x]\Pr[X=x]} \\ & \Pr[Y=x|X=x]\Pr[X=x]\log\frac{\Pr[Y=x]}{\Pr[Y=x|X=x]\Pr[X=x]} \\ & \sum_{y\neq x\wedge y\neq\perp}\Pr[Y=y|X=x]\Pr[X=x]\log\frac{\Pr[Y=y]}{\Pr[Y=y|X=x]\Pr[X=x]} \\ =& \sum_{x}\frac{1-\alpha}{2^n}\log\frac{(1-\alpha)2^n}{1-\alpha} + \frac{\alpha\beta}{2^n}\log\frac{\alpha 2^n}{\alpha\beta 2^n} \\ & + (2^n-1)\frac{\alpha(1-\beta)}{(2^n-1)2^n}\log\frac{\alpha 2^n(2^n-1)}{2^n\alpha(1-\beta)} \\ =& (1-\alpha)n + \alpha\beta\log\frac{1}{\beta} + \alpha(1-\beta)\log\frac{2^n-1}{1-\beta} \\ =& (1-\alpha)n + \alpha((1-\beta)\log(2^n-1) + H(\mathscr{B}_\beta))\end{aligned}$$

Finally, we compute the advantage

$$\begin{aligned}\mathrm{adv}^A &= 1 - \frac{H(X|Y)}{n} \\ &= 1 - (1-\alpha) - \alpha\frac{(1-\beta)\log(2^n-1) + H(\mathscr{B}_\beta)}{n} \\ &= \alpha\left(1 - \frac{(1-\beta)\log(2^n-1) + H(\mathscr{B}_\beta)}{n}\right).\end{aligned}$$

□

Note that for large n we get $\mathrm{adv}^A \approx \alpha^A\beta^A$, which is exactly A's success probability. Plugging this into Definition 28 matches the well-known definition of bit security for search primitives. On the other hand, for $n = 1$ this yields $\mathrm{adv}^A = \alpha^A(1 - H(\mathscr{B}_{\beta^A})) \approx \alpha^A(\delta^A)^2$. This matches the definition of Goldreich and Levin [29, 42], who proposed this definition since it yields the inverse sample complexity of noticing the correlation between the adversary output and the secret. The fact that it can be derived from Definition 27 suggests that this is the "right" definition of the adversary's advantage.

We now redefine the adversary's advantage according to above observations, which, combined with Definition 28 yields the definition of bit security we actually put forward and will use throughout the rest of this work.

Definition 29 *For a search game, the advantage of the adversary A is*

$$\mathrm{adv}^A = \alpha^A\beta^A$$

and for a decision game, it is

$$\mathrm{adv}^A = \alpha^A(\delta^A)^2.$$

Security Reductions

To argue that our definition is useful in a theoretical sense, we apply it to several natural reductions, which arise when constructing cryptographic primitives from other ones. As the novelty of our definition lies mostly with decision games, we will focus on decision primitives that are built from search primitives (cf. Section 5.3.1) and search primitives that are built from decision primitives (cf. Section 5.3.2). Furthermore, in Section 5.3.3 we show that hybrid arguments are valid using our definition, which can be viewed as a reduction between decision primitives. Another such example will follow in Section 6.1.2, where we show a tight reduction from the security of decision primitives using approximate samplers to the security of an idealized primitive with access to the exact distribution.

Throughout this section we will refer to two distribution ensembles $\{\mathscr{D}_0^\theta\}_\theta$ and $\{\mathscr{D}_1^\theta\}_\theta$ as κ-bit indistinguishable, if the indistinguishability game from Definition 25 instantiated with $\{\mathscr{D}_0^\theta\}_\theta$ and $\{\mathscr{D}_1^\theta\}_\theta$ is κ-bit secure.

Search to Decision

A classical way to turn a search primitive into a decision primitive is the Goldreich-Levin hardcore bit[29].

Definition 30 *Let $f : Z \mapsto Y$ be a function and $b : Z \mapsto \{0,1\}$ be a predicate. The* hardcore bit game *is defined as follows: the challenger X picks $z \leftarrow \mathscr{U}(Z)$ and sets the secret to $x \leftarrow b(x)$. It sends $f(z)$ to the adversary which attempts to guess x.*

Goldreich and Levin showed a way to construct a function with a hard-core bit from any one-way function. In this setting, one would hope that if the one-way function is κ-bit secure then also the hard-core bit is close to κ bit secure. The next theorem due to Levin [42] establishes exactly such a connection.

Theorem 6 (adapted from [42]) *Let $f : \{0,1\}^n \mapsto \{0,1\}^k$ be a κ-bit secure one-way function. Then $b(\mathbf{z},\mathbf{r}) = \langle \mathbf{z},\mathbf{r}\rangle \mod 2$ is a $(\kappa - O(\log n))$-bit secure hardcore bit for $g(z,r) = (f(z),r)$.*

This theorem was proven in [42], and all we did was to adapt the statement from [42] to our notation/framework. For completeness, we present the proof in the following.

Proof Throughout the proof the inner product $\langle \cdot,\cdot \rangle$ will be performed implicitly mod 2. We fix any distinguisher D that on input $(f(\mathbf{z}),\mathbf{r})$ outputs $b(\mathbf{z},\mathbf{r})$ with advantage $\alpha^D(\delta^D)^2$. We define the local advantage of D on $\mathbf{z}$ as

$$\alpha_z^D(\delta_z^D)^2 = \Pr[D(f(\mathbf{z}),\mathbf{r}) \neq \perp](2\Pr[D(f(\mathbf{z}),\mathbf{r}) = b(\mathbf{z},\mathbf{r}) \mid D(f(\mathbf{z}),\mathbf{r}) \neq \perp] - 1)^2$$

where the probability is over $\mathbf{r}$ and the internal randomness of D. We will make use of the following fact, which follows from the Cauchy-Schwartz inequality (see [42] for details).

Fact 15 *We have $\alpha^D(\delta^D)^2 \leq \mathbb{E}_{\mathbf{z}\in Z}[\alpha_z^D(\delta_z^D)^2]$.*

We will build an adversary A from D that on input $f(\mathbf{z})$ outputs $\mathbf{z}$ with probability $\alpha_z^D(\delta_z^D)^2$ and otherwise $\perp$. Due to Fact 15 this ensures that $\mathrm{adv}^A \geq \mathrm{adv}^D$. Furthermore, the expected resources T^A of A will be bounded by $\mathrm{poly}(n)T^D$, which will yield the result.

We fix $\mathbf{z}$ and assume that $\delta_z^D > 0$. If A queries $D(f(\mathbf{z}),\mathbf{r})$ for a random $\mathbf{r}$, D will return $b(\mathbf{z},\mathbf{r})$ with probability $\alpha_z^D\beta_z^D$, where $\beta_z^D = (\delta_z^D + 1)/2 > 1/2$. Now let $\mathbf{r}^i$ be $\mathbf{r}$ with the i-th bit flipped and observe that $\langle b(\mathbf{z},\mathbf{r}), b(\mathbf{z},\mathbf{r}^i)\rangle = z_i$ – the i-th bit of $\mathbf{z}$. Accordingly, we have $\langle b(\mathbf{z},\mathbf{r}), D(f(\mathbf{z}),\mathbf{r}^i)\rangle = z_i$ with probability $\alpha_z^D\beta_z^D$ for random $\mathbf{r}$ (note that if $\mathbf{r}$ is random, so is $\mathbf{r}^i$). By Chebyshev's inequality, repeating this process for $2n/\alpha_z^D(\delta_z^D)^2$ pairwise independent $\mathbf{r}$ and taking the majority vote on the non-$\perp$ results from D will result in the correct value of z_i with probability $> 1 - 1/2n$.

Sampling a uniformly random matrix $\mathbf{R} \in \{0,1\}^{n\times k}$ and computing $\mathbf{Rp}$ for all $\mathbf{p} \in \{0,1\}^k \backslash \{\mathbf{0}\}$ results in $2^k - 1$ pairwise random vectors. Let $\mathbf{r} = \mathbf{Rp}$ for some $\mathbf{p}$. Then $b(\mathbf{z},\mathbf{r}) = b(\mathbf{z},\mathbf{Rp}) = b(\mathbf{zR},\mathbf{p})$. Note that $\mathbf{w} = \mathbf{zR} \in \{0,1\}^k$ is unknown to A, but can be guessed.

At this point we have the following procedure: sample $\mathbf{R} \leftarrow \mathscr{U}(\{0,1\}^{n\times k})$ and compute $g_{i,\mathbf{p}} = D(f(\mathbf{z}),(\mathbf{Rp})^i)$ for all $i \in [n]$ and $\mathbf{p} \in \{0,1\}^k \backslash \{\mathbf{0}\}$. Then, for every $\mathbf{w} \in \{0,1\}^k$, compute $\langle b(\mathbf{w},\mathbf{p}), g_{i,\mathbf{p}} \rangle$ for every $\mathbf{p} \in \{0,1\}^k \backslash \{\mathbf{0}\}$ such that $g_{i,\mathbf{p}} \neq \perp$ and set z'_i to the majority of them. Naively, this takes 2^{2k} time, but mapping the domain $\{0,\perp,1\}$ to $\{1,0,-1\}$, the majority can be computed using a summation and taking the sign of the sum. All 2^k summations, i.e. for every $\mathbf{w} \in \{0,1\}^k$, can be computed using the Fast Fourier Transform in time $k2^k$.

If $\mathbf{w} = \mathbf{zR}$, then $\mathbf{z}' = \mathbf{z}$ with probability $> 1/2$ by union bound, assuming that $2^k - 1 \geq 2n/\alpha_z^D(\delta_z^D)^2$. The correctness of $\mathbf{z}'$ can be checked by computing $f(\mathbf{z}')$. To accommodate for the assumption that $\delta^D > 0$, one also checks $f(\bar{\mathbf{z}}')$, where $\bar{\mathbf{z}}'$ is obtained by flipping all the bits of $\mathbf{z}'$.

Finally, A, on input $f(\mathbf{z})$, samples $l \leq 2n$ bits from $\mathscr{U}(\{0,1\})$ until the first 0 (if no 0 occurs, simply output $\perp$). Then it calls above procedure with $k = l + \lceil \log 5n \rceil$. Its

success probability is $1/2$ times the probability that $k \geq \log(2n/\alpha_z^D(\delta_z^D)^2+1)$:

$$\begin{aligned}
\Pr[k \geq \log(2n/\alpha_z^D(\delta_z^D)^2+1)] &= \sum_{k=\lceil \log(2n/\alpha_z^D(\delta_z^D)^2+1)\rceil}^{2n} p_k \\
&= \sum_{k=\lceil \log(2n/\alpha_z^D(\delta_z^D)^2+1)\rceil}^{2n} 2^{-k+\lceil \log 5n\rceil} \\
&\geq 5n \sum_{k=\lceil \log(2n/\alpha_z^D(\delta_z^D)^2+1)\rceil}^{2n} 2^{-k} \\
&\geq 10n \left(\frac{\alpha_z^D(\delta_z^D)^2}{8n} - 2^{-2n-1} \right) \\
&\gtrsim \alpha_z^D(\delta_z^D)^2.
\end{aligned}$$

Note that the small additive factor $O(n2^{-2n-1})$ hidden in the last inequality leads to a a minuscule loss in security and thus covered by the $O(\log \kappa)$ term claimed in the theorem.

The expected running time is bounded by

$$\begin{aligned}
\sum_{k=\lceil \log 5n\rceil}^{2n+\lceil \log 5n\rceil} p_k k 2^k \mathrm{poly}(n) T^D &= \mathrm{poly}(n) T^D \sum_{k=\lceil \log 5n\rceil}^{2n+\lceil \log 5n\rceil} 2^{-k+\lceil \log 5n\rceil} k 2^k \\
&= \mathrm{poly}(n) T^D \sum_{k=\lceil \log 5n\rceil}^{2n+\lceil \log 5n\rceil} k \\
&= \mathrm{poly}(n) T^D.
\end{aligned}$$

□

The proof for this theorem assumes a distinguisher D for b and constructs from it an inverter A for f, where $\mathrm{adv}^D = \mathrm{adv}^A$ (and the running time is polynomially related). Such security preserving reductions are information theoretically only possible with a definition of advantage that is proportional to $(\delta^D)^2$ for decision primitives, if it is proportional to $\alpha^A\beta^A$ for search primitives. This is because any inverter querying a

distinguisher with advantage δ^D and attempting to learn an $\alpha^A\beta^A$ fraction of a secret with at least one bit of entropy, must make at least $\Omega(\alpha^A\beta^A/(\delta^D)^2)$ queries. Denote the resources of D by T^D and note that $T^A \geq \Omega(\alpha^A\beta^A/(\delta^D)^2)T^D$ is a lower bound on the resources of A. The goal of the proof is to find an upper bound on $T^A/\mathrm{adv}^A = T^A/\alpha^A\beta^A \geq \Omega(T^D/(\delta^D)^2)$. This is only possible by assuming an upper bound on $T^D/(\delta^D)^2$. If only a bound on T^D/δ^D is assumed, then the upper bound on T^A/adv^A must contain a linear factor in $1/\delta^D$, which may be as large as $O(2^n)$ and thus result in a dramatic loss in (nominal) security.

Decision to Search

In the following subsections we show constructions and the corresponding reductions in the other direction. The first is just a straightforward converse to the Goldreich-Levin theorem, showing that any PRG is also a OWF for the same bit security. The second construction is presented as a very natural and straight-forward way of turning a decision primitive into a search primitive. The third reduction is one that naturally arises in cryptographic applications, for example identification protocols.

PRGs are one-way functions

While the following theorem is intuitively trivial (and technically simple), as explained in the introduction it serves to justify our definition of bit security. The proof also illustrates the subtle difference between an adversary that outputs $\perp$ and one that outputs a random guess.

Theorem 7 *If g is a PRG with κ-bit security, then it is also a $(\kappa-4)$-bit secure one-way function.*

Proof Assume A is an attack to g as a one-way function with cost T, output probability α^A, and conditional success probability β^A. We turn A into an adversary D to g as a PRG

by letting $D(y)$ output 1 if $G(A(y)) = y$ and $\perp$ otherwise. Assume that A has conditional success probability $\beta^A = 1$. This is without loss of generality because one-way function inversion is a verifiable search problem, and A can be modified (without affecting its advantage) to output $\perp$ when its answer is incorrect. So, A has advantage α^A, equal to its output probability. Notice that D is successful only when the indistinguishability game chooses the secret bit 1, and then A correctly inverts the PRG. So, the success probability of D is precisely $\alpha^D\beta^D = \alpha^A/2$. The output probability of D can be a bit higher, to take into account the possibility that on secret bit 0, the challenger picks a random string that belongs (by chance) to the image of the PRG, and A correctly inverts it. But, in any case, it always belongs to the interval $\alpha^D \in [1/2, 3/4] \cdot \alpha^A$. It follows that $\alpha^D \geq \alpha^A/2$ and $\beta^D = (\alpha^A/2)/\alpha^D \geq 2/3$. So, D has advantage at least $\alpha^D(\delta^D)^2 = \alpha^D(2\beta^D - 1)^2 \geq \alpha^A/9$. Since the two algorithms have essentially the same cost, they achieve the same level of bit security, up to a small constant additive term $\log 9 < 4$. $\square$

We remark that our proof differs from the standard text-book reduction that pseudorandom generators are one-way functions in a simple, but crucial way: when $A(y)$ fails to invert G, instead of outputting 0 as a "best guess" at the decision problem, it outputs $\perp$ to explicitly declare failure. The reader can easily check that the standard reduction has output probability $\alpha^D = 1$ and (conditional) success probability $\beta^D \leq (\alpha^A + 1)/2$. So, the advantage of the distinguisher in the standard proof is $\alpha^D(2\beta^D - 1)^2 = (\alpha^A)^2$, resulting in a substantial drop ($\log \alpha^A$) in the bit security proved by the reduction.

Secret Recovery

We proceed by giving a construction of a search primitive from two distributions. We are not aware of any immediate applications, but this simple example is supposed to serve as evidence that our definitions for search and decision primitives behave nicely under composition. It also provides an example of "non verifiable" search problem, i.e., a cryptographic problem with exponentially large secret space defined by a game at the end of which A cannot efficiently determine if the secret has been found. Differently from Theorem 7, this time one *cannot* assume without loss of generality that the (hypothetical) attacker to the search problem has conditional success probability $\beta = 1$.

Definition 31 *Let $\mathscr{D}_0, \mathscr{D}_1$ be two distributions. We define the n-bit* secret recovery game *as the following n-bit security game: the challenger X chooses an n-bit secret $x \leftarrow \mathscr{U}(\{0,1\}^n)$ and sends the vector $\mathbf{c} = (c_i \leftarrow D_{x_i})_{i \leq n}$ to A. The adversary A attempts to guess x, i.e. R is the equality relation.*

The next theorem shows that when instantiating the game with two indistinguishable distributions, the secret recovery game enjoys essentially the same bit security.

Theorem 8 *If the κ-bit secret recovery game is instantiated with two κ-bit secure indistinguishable distributions $\mathscr{D}_0$ and $\mathscr{D}_1$, and $\mathscr{D}_0$ is publicly sampleable, then it is $(\kappa - 1)$-bit secure.*

Proof Let A be an adversary against the secret recovery game that recovers x from the vector $\mathbf{c}$ with advantage $\mathrm{adv}^A = \alpha^A \beta^A$. We build a distinguisher D against the indistinguishability of $\mathscr{D}_0$ and $\mathscr{D}_1$ with essentially the same resources and advantage: D chooses a secret $x \in \{0,1\}^\kappa$ uniformly at random, which is non-zero with high probability (otherwise output $\perp$) and constructs the vector $\mathbf{c}$ by sampling $\mathscr{D}_0$ itself for every zero bit

in x and querying its oracle for every 1 bit in x (which will return either samples from $\mathscr{D}_0$ or from $\mathscr{D}_1$). It sends $\mathbf{c}$ to A and returns 1 iff A returns x, otherwise it outputs $\perp$.

The resources of D are essentially the same as those of A, so we analyze its advantage $\text{adv}^D = \alpha^D(\delta^D)^2$. The output probability of D, conditioned on $x \neq 0$, is almost exactly A's success probability, but note that A is only presented with the correct input distribution if D's challenger returns samples from $\mathscr{D}_1$, which is the case with probability $\frac{1}{2}$. So $\alpha^D \geq \frac{1-2^{-\kappa}}{2}\alpha^A\beta^A$. Furthermore, D's conditional distinguishing advantage is $\delta^D \geq 1-2^{-\kappa+1}$, because it only outputs the incorrect value if A returned x even though $\mathbf{c}$ consisted of samples only from $\mathscr{D}_0$. Note that in this case A has no information about x, which was chosen uniformly at random and thus the probability of this event is at most $2^{-\kappa}$. Accordingly, $\text{adv}^D = \alpha_D(\delta^D)^2 \geq \frac{(1-2^{-\kappa+1})^2}{2}\alpha^A\beta^A \approx \text{adv}_A/2$. □

Indistinguishability implies Message-Hiding

In our last example for this section we show that IND-CCA secure encryption schemes enjoy a message hiding property, which we first formally define.

Definition 32 *A private or public key encryption scheme is κ-bit* message hiding*, if the following security game is κ-bit secure: the challenger chooses a message $m \in \{0,1\}^n$ uniformly at random and sends its encryption to A. The adversary A attempts to guess m, while C provides it with encryption (in case of private key schemes) and decryption oracles.*

This property naturally arises in the context of constructions of identification protocols from encryption schemes (see e.g. [6]), where a random message is encrypted and identification relies on the fact that only the correct entity can decrypt it. While it seems intuitively obvious that breaking message hiding is no easier than distinguishing

encrypted messages, showing that this is true in a quantifiable sense for specific definitions of bit security is not as obvious. The next theorem establishes this connection.

Theorem 9 *If a scheme with message space larger than* 2^κ *is* κ*-bit IND-CCA secure, it is* κ*-bit* message hiding.

Proof Let A be an adversary that is able to extract a random message from an encryption scheme with advantage $\text{adv}^A = \alpha^A \beta^A$. We construct a IND-CCA distinguisher D against the scheme with essentially the same resources and advantage: D generates two messages $m_0, m_1 \leftarrow \{0,1\}^m$ uniformly at random, which are distinct with overwhelming probability (if not, output $\perp$). It sends them to the challenger, which encrypts one of them. Upon receiving the challenge cipher text c_b, D forwards it to A. Any queries to the encryption (in case of private key encryption) or decryption oracle are simply forwarded to D's own oracles. If A returns a message in $\{m_0, m_1\}$, D returns the corresponding bit. Otherwise, it outputs $\perp$.

The resources of D are essentially the same as for A, so we focus on its advantage. Note that conditioned on the event that $m_0 \neq m_1$, D's output probability α^D is at least as large as the success probability of A, so $\alpha^D \geq (1-2^{-\kappa})\alpha^A\beta^A$. The conditional distinguishing advantage of D is $\delta^D \geq 1-2^{-\kappa+1}$, since the only way D will guess incorrectly is when A somehow outputs the wrong message $m_{\bar{b}}$. Since A has no information about this message (which was chosen uniformly at random), the probability of this happening is at most $2^{-\kappa}$. This shows that D's advantage in the indistinguishability game is $\text{adv}^D = \alpha^D(\delta^D)^2 \geq (1-2^{-\kappa})\alpha^A\beta^A(1-2^{-\kappa+1})^2 \approx \alpha^A\beta^A = \text{adv}^A$, where the latter is A's advantage in the message hiding game. $\square$

Decision to Decision – The Hybrid Argument

This section is devoted to proving a general hybrid argument for indistinguishability games using our definition of advantage. Formally, we prove the following lemma.

Lemma 11 *Let $\mathscr{H}_i$ be k distributions and $G_{i,j}$ be the indistinguishability game instantiated with $\mathscr{H}_i$ and $\mathscr{H}_j$. Further, let $\varepsilon_{i,j} = \max_A \mathrm{adv}^A$ over all T-bounded adversaries A against $G_{i,j}$. Then $\varepsilon_{1,k} \leq 3k \sum_i^{k-1} \varepsilon_{i,i+1}$.*

Applying the lemma to our definition of bit security, we immediately get the following theorem.

Theorem 10 *Let $\mathscr{H}_i$ be k distributions. If $\mathscr{H}_i$ and $\mathscr{H}_{i+1}$ are κ-bit indistinguishable for all i, then $\mathscr{H}_1$ and $\mathscr{H}_k$ are $(\kappa - 2(\log k + 1))$-bit indistinguishable.*

Proof Let A be any adversary with resources T^A (when attacking $\mathscr{H}_1$ and $\mathscr{H}_k$). By assumption, $\varepsilon_{i,i+1} \leq T^A/2^\kappa$ (where $\varepsilon_{i,j}$ is defined as in Lemma 11) for all T^A-bounded adversaries against $\mathscr{H}_i$ and $\mathscr{H}_{i+1}$. By Lemma 11, $\varepsilon_{i,k} \leq 3k^2 T^A/2^\kappa$ for all T^A-bounded adversaries, in particular A. □

As a simple application, we get the following corollary.

Corollary 5 *If a public key encryption scheme is κ-bits IND-CCA secure, then it is $(\kappa - 2(\log k + 1))$-bit IND-CCA secure in the k message setting.*

Note that in contrast to the standard hybrid argument, which simply exploits the triangle inequality of statistical distance, we lose an additional factor of $3k$ in the advantage. In particular, consider the case where the bounds $\varepsilon = \varepsilon_{i,i+1}$ are the same for all i. This means that $\varepsilon_{1,k} \leq 3k^2\varepsilon$. Note that this additional factor has only a minor impact on bit security. (See below for details.) Still, one may wonder if this additional factor is an

artifact of a non-tight proof or if it is indeed necessary. Consider any distinguishers D that never outputs $\perp$ (i.e. $\alpha^D = 1$). The distinguishing advantage $\delta^D_{i,j}$ against distributions $\mathscr{H}_i$ and $\mathscr{H}_j$ is exactly the statistical distance between $D(\mathscr{H}_i)$ and $D(\mathscr{H}_j)$. Assume $\delta^D_{i,i+1} \leq \varepsilon$ for all i. The standard hybrid argument implies that $\delta^D_{1,k}$ cannot be larger than – but may be as large as – $k\varepsilon$. But this can provide as much as $H(\mathscr{B}_{\frac{1}{2}+k\varepsilon}) \approx (k\varepsilon)^2$ bits of information about the secret bit. Applying the same argument to any indistinguishability game instantiated with $\mathscr{H}_i$ and $\mathscr{H}_{i+1}$ shows that D provides at most ε^2 bits of information about the secret bit by assumption. This shows that the information learned by D is a factor k^2 larger on $\mathscr{H}_1$ and $\mathscr{H}_k$! Since this is the intuition we base our definition on, we believe that either the standard hybrid argument is not tight, or our theorem is tight (up to the constant factor 3). Either way, as Theorem 10 and Corollary 5 demonstrate, this additional factor only affects the constant in front of the log term in the number of hybrids, so, we believe, it is only of secondary importance.

The rest of the subsection proves Lemma 11, where we make use of the following notation. For some distinguisher D, let $\alpha^D_{\mathscr{P},\mathscr{Q}}$ be its output probability, $\beta^D_{\mathscr{P},\mathscr{Q}}$ its conditional success probability, $\delta^D_{\mathscr{P},\mathscr{Q}}$ its conditional distinguishing advantage, and $\mathrm{adv}^D_{\mathscr{P},\mathscr{Q}} = \alpha^D_{\mathscr{P},\mathscr{Q}}(\delta^D_{\mathscr{P},\mathscr{Q}})^2$ its advantage against the distributions $\mathscr{P}, \mathscr{Q}$. Furthermore, let $\alpha^D_{\mathscr{P}} = \Pr[D(\mathscr{P}) \neq \perp]$ and $\gamma^D_{\mathscr{P}} = \Pr[D(\mathscr{P}) = 1]$ for any distribution $\mathscr{P}$. We can express the advantage of D against $\mathscr{P}$ and $\mathscr{Q}$ in terms of $\alpha^D_{\mathscr{P}}$, $\alpha^D_{\mathscr{Q}}$, $\gamma^D_{\mathscr{P}}$, $\gamma^D_{\mathscr{Q}}$:

$$\begin{aligned}
\alpha^D_{\mathscr{P},\mathscr{Q}} &= \frac{1}{2}(\alpha^D_{\mathscr{P}} + \alpha^D_{\mathscr{Q}}) \\
\beta^D_{\mathscr{P},\mathscr{Q}} &= \frac{\gamma^D_{\mathscr{P}} - \gamma^D_{\mathscr{Q}} + \alpha^D_{\mathscr{Q}}}{\alpha^D_{\mathscr{P}} + \alpha^D_{\mathscr{Q}}} \\
\delta^D_{\mathscr{P},\mathscr{Q}} &= 2\beta^D_{\mathscr{P},\mathscr{Q}} - 1 = \frac{2(\gamma^D_{\mathscr{P}} - \gamma^D_{\mathscr{Q}}) + \alpha^D_{\mathscr{Q}} - \alpha^D_{\mathscr{P}}}{\alpha^D_{\mathscr{P}} + \alpha^D_{\mathscr{Q}}} \\
\mathrm{adv}^D_{\mathscr{P},\mathscr{Q}} &= \frac{(2(\gamma^D_{\mathscr{P}} - \gamma^D_{\mathscr{Q}}) + \alpha^D_{\mathscr{Q}} - \alpha^D_{\mathscr{P}})^2}{2(\alpha^D_{\mathscr{P}} + \alpha^D_{\mathscr{Q}})}.
\end{aligned} \tag{5.2}$$

We begin with the observation that for computationally indistinguishable distributions the output probabilities of any bounded distinguisher D cannot vary too much under the two distributions.

Lemma 12 *Let $\mathscr{P}, \mathscr{Q}$ be two distributions. If $\mathrm{adv}^D_{\mathscr{P},\mathscr{Q}} \leq \varepsilon$ for all T-bounded distinguishers, then we have $\alpha^D_{\mathscr{P}} \leq 2\alpha^D_{\mathscr{Q}} + 3\varepsilon$ and $\alpha^D_{\mathscr{Q}} \leq 2\alpha^D_{\mathscr{P}} + 3\varepsilon$ for any T bounded distinguisher.*

Proof We prove the first claim. (The proof of the second claim is symmetrical.) Fix any distinguisher D. Assume $\alpha^D_{\mathscr{P}} \geq 2\alpha^D_{\mathscr{Q}}$, since otherwise we are done. Consider an alternative distinguisher D', which runs D and in the event that $D \neq \perp$, outputs 1 and otherwise $\perp$. Obviously, D' is also T-bounded, and (setting $\gamma^{D'}_{\mathscr{P}} = \alpha^{D'}_{\mathscr{P}}$, $\gamma^{D'}_{\mathscr{Q}} = \alpha^{D'}_{\mathscr{Q}}$ in (5.2)) we get

$$\begin{aligned}
\mathrm{adv}^{D'}_{\mathscr{P},\mathscr{Q}} &= \frac{(\alpha^D_{\mathscr{P}} - \alpha^D_{\mathscr{Q}})^2}{2(\alpha^D_{\mathscr{P}} + \alpha^D_{\mathscr{Q}})} \\
&\geq \frac{(\alpha^D_{\mathscr{P}} - \alpha^D_{\mathscr{Q}})^2}{3\alpha^D_{\mathscr{P}}} \\
&= \frac{1}{3}\left(\alpha^D_{\mathscr{P}} - 2\alpha^D_{\mathscr{Q}} + \frac{(\alpha^D_{\mathscr{Q}})^2}{\alpha^D_{\mathscr{P}}}\right) \\
&\geq \frac{1}{3}\left(\alpha^D_{\mathscr{P}} - 2\alpha^D_{\mathscr{Q}}\right).
\end{aligned}$$

The first claim now follows from $\varepsilon \geq \mathrm{adv}^{D'}_{\mathscr{P},\mathscr{Q}}$. $\square$

Proof [of Lemma 11] We fix any distinguisher D and drop the superfix of α, γ, δ and adv for the rest of the proof. Furthermore, we will abbreviate $\mathscr{H}_i$ by i in the subfixes of α, γ, δ, and adv.

Using induction, one can prove

$$\sum_{i=1}^{k} \mathrm{adv}_{i,i+1} \geq \frac{\alpha_1 + \alpha_k}{\alpha_1 + 2\sum_{i=2}^{k-1} \alpha_i + \alpha_k} \mathrm{adv}_{1,k}$$

The proof proceeds by substituting in the definition of $\mathrm{adv}_{i,i+1}$ from (5.2), applying the induction hypothesis to the first $k-1$ terms of the sum, and then minimizing over γ_{k-1}.

It remains to show that

$$\frac{\alpha_1 + \alpha_k}{\alpha_1 + 2\sum_{i=2}^{k-1} \alpha_i + \alpha_k} \geq \frac{1}{3k}.$$

We again proceed by induction and can thus assume that $\mathrm{adv}_{1,i} \leq 3i\sum_{j=1}^{i-1} \varepsilon_{j,j+1}$ for all $i < k$ and symmetrically $\mathrm{adv}_{i,k} \leq 3(k-i)\sum_{j=i}^{k-1} \varepsilon_{j,j+1}$ for all $i > 1$. By Lemma 12, this means that $\alpha_i \leq 2\alpha_1 + 9i\sum_{j=1}^{i-1} \varepsilon_{j,j+1}$ for all $i < k$ and again $\alpha_i \leq 2\alpha_k + 9(k-i)\sum_{j=i}^{k-1} \varepsilon_{j,j+1}$ for all $i > 1$. We note that

$$\alpha_1 + 2\sum_{i=2}^{k-1} \alpha_i + \alpha_k = \alpha_1 + 2\sum_{i=2}^{\lfloor (k-1)/2 \rfloor} \alpha_i + 2\sum_{\lfloor (k-1)/2 \rfloor + 1}^{k-1} \alpha_i + \alpha_k$$

and using the above inequalities, the two sums are bounded by

$$2\sum_{i=2}^{\lfloor (k-1)/2 \rfloor} \alpha_i \leq 2(k-3)\alpha_1 + 3k^2 \sum_{i=1}^{\lfloor (k-1)/2 \rfloor} \varepsilon_{i,i+1}$$

and

$$2\sum_{\lfloor (k-1)/2\rfloor+1}^{k-1}\alpha_i \leq 2(k-3)\alpha_k + 3k^2\sum_{\lfloor (k-1)/2\rfloor+1}^{k-1}\varepsilon_{i,i+1}$$

respectively. This bounds the entire sum:

$$\alpha_1 + 2\sum_{i=2}^{k-1}\alpha_i + \alpha_k \leq 2k(\alpha_1+\alpha_k) + 3k^2\sum_{i=1}^{k-1}\varepsilon_{i,i+1}$$

This in turn leads to the lower bound

$$\frac{\alpha_1+\alpha_k}{\alpha_1 + 2\sum_{i=2}^{k-1}\alpha_i + \alpha_k} \geq \frac{1}{2k + \frac{3k^2\sum_{i=1}^{k-1}\varepsilon_{i,i+1}}{\alpha_1+\alpha_k}}$$

The last step is noticing that we can assume that $(\alpha_1+\alpha_k) \geq 6k\sum_{i=1}^{k-1}\varepsilon_{i,i+1}$, because $(\alpha_1+\alpha_k)/2 \geq \varepsilon_{1,k}$ and otherwise we would be done. Using this assumption we have

$$\frac{\alpha_1+\alpha_k}{\alpha_1 + 2\sum_{i=2}^{k-1}\alpha_i + \alpha_k} \geq \frac{1}{2k + \frac{3k^2}{6k}} \geq \frac{1}{3k}$$

as desired. □

Chapter 5, in full, is a reprint of material (with minor modifications) that has been submitted for publication and may appear as "On the Bit Security of Cryptographic Primitives" by Daniele Micciancio and Michael Walter. The dissertation author was the primary investigator and author of this paper.

Chapter 6

Approximate Samplers

The Security of Approximate Samplers

Many security reductions for lattice-based cryptographic primitives assume that the primitive has access to samplers for an ideal distribution, which may be too difficult or costly to sample from, and is routinely replaced by an approximation in any concrete implementation. Naturally, if the approximation is good enough, then security with respect to the ideal distribution implies that the actual implementation (using the approximate distribution) is also secure. But evaluating how the quality of approximation directly affects the concrete security level achieved by the primitive can be a rather technical task. Traditionally, the quality of the approximation has been measured in terms of the statistical distance $\delta = \Delta_{\mathrm{SD}}$, which satisfies the following useful properties:

1. *Probability preservation:* for any event E over the random variable X we have $Pr_{X \leftarrow \mathscr{P}}[E] \geq Pr_{X \leftarrow \mathscr{Q}}[E] - \delta(\mathscr{P}, \mathscr{Q})$. This property allows to bound the probability of an event occurring under $\mathscr{P}$ in terms of the probability of the same event occurring under $\mathscr{Q}$ and the quantity $\delta(\mathscr{P}, \mathscr{Q})$. It is easy to see that this property is equivalent to the bound $\Delta_{\mathrm{SD}}(\mathscr{P}, \mathscr{Q}) \leq \delta(\mathscr{P}, \mathscr{Q})$. So the statistical distance $\delta = \Delta_{\mathrm{SD}}$ satisfies this property by definition.

2. *Sub-additivity for joint distributions:* if $(X_i)_i$ and $(Y_i)_i$ are two lists of discrete

random variables over the support $\prod_i S_i$, then

$$\delta((X_i)_i,(Y_i)_i) \leq \sum_i \max_a \delta([X_i \mid X_{<i} = a],[Y_i \mid Y_{<i} = a]),$$

where $X_{<i} = (X_1,\ldots,X_{i-1})$ (and similarly for $Y_{<i}$), and the maximum is taken over $a \in \prod_{j<i} S_j$.

3. *Data processing inequality:* $\delta(f(\mathscr{P}),f(\mathscr{Q})) \leq \delta(\mathscr{P},\mathscr{Q})$ for any two distributions $\mathscr{P}$ and $\mathscr{Q}$ and (possibly randomized) algorithm $f(\cdot)$, i.e., the measure does not increase under function application.

We call any divergence that satisfies these three properties a *useful* divergence.

Approximate Samplers and Search Primitives

The following simple lemma demonstrates a classical proof of security using a useful divergence, where, for simplicity, we assume that the resources T_A of the adversary do not depend on the distributions $\mathscr{P}_\theta$, and that the number of calls to $\mathscr{P}_\theta$ performed during any run of the game $G^{\mathscr{P}}_{S,A}$ is bounded from above by T_A.

Lemma 13 *Let $S^{\mathscr{P}}$ be a scheme with black-box access to a probability distribution ensemble $\mathscr{P}_\theta$. Let its security against adversary A be defined by a* search *game $G^{\mathscr{P}}_{S,A}$ and , and δ any cryptographically useful divergence. If $S^{\mathscr{P}}$ is κ-bit secure and $\delta(\mathscr{P}_\theta,\mathscr{Q}_\theta) \leq 2^{-\kappa}$, then $S^{\mathscr{Q}}$ is $(\kappa-1)$-bit secure.*

Before we prove Lemma 13, we begin with a technical observation that will be used throughout this section.

Lemma 14 *Let δ be a cryptographically useful divergence, let $\mathscr{P}_\theta$ and $\mathscr{Q}_\theta$ be two probability ensembles and let $A^{\mathscr{P}}$ be an algorithm querying $\mathscr{P}_\theta$ at most q times. Let θ_i*

(resp. $\tilde{\theta}_i$) be the distribution of the i-th query made by $A^{\mathscr{P}}$ (resp. $A^{\mathscr{Q}}$). Then,

$$\delta((\theta_i, \mathscr{P}_{\theta_i} \mid X_i), (\tilde{\theta}_i, \mathscr{Q}_{\tilde{\theta}_i} \mid X_i)) \leq \delta(\mathscr{P}_\theta, \mathscr{Q}_\theta)$$

for any event X_i that $(\theta_j, \mathscr{P}_{\theta_j})_{j<i}$ and $(\tilde{\theta}_j, \mathscr{Q}_{\tilde{\theta}_j})_{j<i}$ take some specific (and identical) value.

Proof Note that at any point during the execution of A, conditioned on the event X_i, $A^{\mathscr{P}}$ and $A^{\mathscr{Q}}$ behave identically up to the point they make the ith query. In particular, the conditional distributions $(\theta_i \mid X_i)$ and $(\tilde{\theta}_i \mid X_i)$ are identical and $\delta((\theta_i \mid X_i), (\tilde{\theta}_i \mid X_i)) = 0$. It follows by subadditivity (for joint distributions) that

$$\begin{aligned}\delta((\theta_i, \mathscr{P}_{\theta_i} \mid X_i), (\tilde{\theta}_i, \mathscr{Q}_{\tilde{\theta}_i} \mid X_i)) &\leq \delta((\theta_i \mid X_i), (\tilde{\theta}_i \mid X_i)) + \max_\theta \delta(\mathscr{P}_\theta, \mathscr{Q}_\theta) \\ &= \max_\theta \delta(\mathscr{P}_\theta, \mathscr{Q}_\theta).\end{aligned}$$

□

Proof [*of Lemma 13*] Fix any adversary A. First observe that the number of queries drawn from $\mathscr{P}_\theta$ ($\mathscr{Q}_\theta$ resp.) is bounded by the resources T^A and define $\varepsilon_A^{\mathscr{D}} = \mathrm{adv}_A$ to be the advantage of A against $S^{\mathscr{D}}$ for $\mathscr{D} \in \{\mathscr{P}, \mathscr{Q}\}$. Note that $\varepsilon_A^{\mathscr{D}}$ is simply A's success probability, since we are only considering search games. Then we have

$$\varepsilon_A^{\mathscr{P}} \geq \varepsilon_A^{\mathscr{Q}} - \delta(G_{S,A}^{\mathscr{P}}, G_{S,A}^{\mathscr{Q}})$$

by probability preservation of δ. By the bound on the number of queries and the data processing inequality, we have $\delta(G_{S,A}^{\mathscr{P}}, G_{S,A}^{\mathscr{Q}}) \leq \delta((\theta_i, \mathscr{P}_{\theta_i})_{i<T^A}, (\tilde{\theta}_i, \mathscr{Q}_{\tilde{\theta}_i})_{i<T^A})$, where $(\theta_i)_{i<T^A}$ (resp. $(\tilde{\theta}_i)_{i<T^A}$) is the sequence of queries made to $\mathscr{P}_\theta$ (resp. $\mathscr{Q}_\theta$) when A

is attacking $S^{\mathscr{P}}$ (resp. $S^{\mathscr{Q}}$). By sub-additivity for joint distributions and Lemma 14, this quantity is at most $T^A\delta(\mathscr{P}_\theta,\mathscr{Q}_\theta)$. So $\varepsilon_A^{\mathscr{P}} \geq \varepsilon_A^{\mathscr{Q}} - T^A\delta(\mathscr{P}_\theta,\mathscr{Q}_\theta)$. Dividing by T^A, we get $\frac{\varepsilon_A^{\mathscr{P}}}{T^A} \geq \frac{\varepsilon_A^{\mathscr{Q}}}{T^A} - \delta(\mathscr{P}_\theta,\mathscr{Q}_\theta) \geq \frac{\varepsilon_A^{\mathscr{Q}}}{T^A} - 2^{-\kappa}$. Because $S^{\mathscr{P}}$ is κ bit secure, we have $2^{-\kappa} \geq \frac{\varepsilon_A^{\mathscr{P}}}{T^A} \geq \frac{\varepsilon_A^{\mathscr{Q}}}{T^A} - 2^{-\kappa}$, or, equivalently, $2^{1-\kappa} \geq \frac{\varepsilon_A^{\mathscr{Q}}}{T^A}$. This shows that $\log \frac{T^A}{\varepsilon_A^{\mathscr{Q}}} \geq \kappa - 1$, i.e., $S^{\mathscr{Q}}$ provides $\kappa - 1$ bits of security. □

Lemma 13 captures the intuition that security with respect to an ideal distribution implies security with respect to any sufficiently good approximation, and it also gives a way to establish concrete security bounds. In order to (almost) preserve κ bits of security, one needs $\delta(\mathscr{P}_\theta,\mathscr{Q}_\theta) < 2^{-\kappa}$, e.g., as obtained, using $\delta = \Delta_{\mathrm{SD}}$ and estimating the ideal probabilities $\mathscr{Q}(x)$ with κ-bit (fixed point or floating point) approximations. Additionally, Lemma 13 allows us to view $\mathscr{D}_{\mathbb{Z},c,s}$ as a ts-bounded distribution without losing security. Notice that for a security parameter κ we can set t to about $\sqrt{\kappa \ln 2/\pi} \approx \eta_{2^{-\kappa}}(\mathbb{Z})$, which by Lemma 2 implies a statistical distance of less than $2^{-\kappa}$ if $s \geq \eta_\varepsilon(\mathbb{Z})$. So in the rest of this work we will identify the unbounded Gaussian distribution $\mathscr{D}_{\mathbb{Z},c,s}$ with its truncation with support $\mathbb{Z} \cap [c \pm ts]$ whenever appropriate.

While using Δ_{SD} is asymptotically efficient, it has been observed that in practice it can lead to unnecessarily large memory cost and slow computations. The work of [65] showed that we can improve the security analysis of approximate distributions. Assume we have a divergence δ that satisfies the following strengthening of the probability preservation property:

1.* *Pythagorean probability preservation* with parameter $\lambda \in \mathbb{R}$, which states that for any joint distributions $(\mathscr{P}_i)_i$ and $(\mathscr{Q}_i)_i$ over support $\prod_i S_i$, if

$$\delta(\mathscr{P}_i \mid a_i, \mathscr{Q}_i \mid a_i) \leq \lambda$$

for all i and $a_i \in \prod_{j<i} S_j$, then

$$\Delta_{\mathrm{SD}}((\mathscr{P}_i)_i, (\mathscr{Q}_i)_i) \leq \|(\max_{a_i} \delta(\mathscr{P}_i \mid a_i, \mathscr{Q}_i \mid a_i))_i\|_2.$$

We call a divergence that satisfies this property λ-*pythagorean*. A pythagorean divergence additionally satisfying sub-additivity for joint distributions and the data processing inequality (i.e. properties 2 and 3) will be called λ-*efficient*. Using a pythagorean δ, we can improve Lemma 13 as follows.

Lemma 15 *Let $S^{\mathscr{P}}$ be a scheme with black-box access to a probability distribution ensemble $\mathscr{P}_\theta$. Let its security against adversary A be defined by a* search *game $G^{\mathscr{P}}_{S,A}$. If $S^{\mathscr{P}}$ is κ-bit secure and $\delta(\mathscr{P}_\theta, \mathscr{Q}_\theta) \leq 2^{-\kappa/2}$ for some $2^{-\kappa/2}$-*efficient *divergence δ, then $S^{\mathscr{Q}}$ is $(\kappa-3)$-bit secure.*

Proof Fix any adversary A. Define $\varepsilon^{\mathscr{D}}_A = \mathrm{adv}_A$ to be the advantage of A against $S^{\mathscr{D}}$ for $\mathscr{D} \in \{\mathscr{P}, \mathscr{Q}\}$. Towards a contradiction, assume we have $\frac{T^A}{\varepsilon^{\mathscr{P}}_A} \geq 2^\kappa$, but $\frac{T^A}{\varepsilon^{\mathscr{Q}}_A} < 2^{\kappa-3}$. Consider the hypothetical game $[G^{\mathscr{Q}}_{S,A}]^n$ (resp. $[G^{\mathscr{P}}_{S,A}]^n$) consisting of n independent copies of $G^{\mathscr{Q}}_{S,A}$ (resp. $G^{\mathscr{P}}_{S,A}$). Denote the probability of the event that A wins at least one of the n games by $\varepsilon^{\mathscr{Q}}_{A^n}$ (resp. $\varepsilon^{\mathscr{P}}_{A^n}$). We begin by showing that we can bound $\varepsilon^{\mathscr{P}}_{A^n}$ from below in terms of $\varepsilon^{\mathscr{Q}}_{A^n}$ using probability preservation and data processing inequality of Δ_{SD}:

$$\varepsilon^{\mathscr{P}}_{A^n} \geq \varepsilon^{\mathscr{Q}}_{A^n} - \Delta_{SD}([G^{\mathscr{P}}_{S,A}]^n, [G^{\mathscr{Q}}_{S,A}]^n) \geq \varepsilon^{\mathscr{Q}}_{A^n} - \Delta_{SD}((\theta_i, \mathscr{P}_{\theta_i})_i, (\tilde{\theta}_i, \mathscr{Q}_{\tilde{\theta}_i})_i)$$

where $(\theta_i)_i$ (resp. $(\tilde{\theta}_i)_i$) is the sequence of queries made during the game $[G^{\mathscr{P}}_{S,A}]^n$ (resp. $[G^{\mathscr{Q}}_{S,A}]^n$).

Lemma 14 ensures that we can apply pythagorean probability preservation (Prop-

erty 1*) to obtain

$$\varepsilon_{A^n}^{\mathscr{P}} \geq \varepsilon_{A^n}^{\mathscr{Q}} - \sqrt{T^{A^n}} \cdot \delta(\mathscr{P}_\theta, \mathscr{Q}_\theta) \geq \varepsilon_{A^n}^{\mathscr{Q}} - \sqrt{T^{A^n}} \cdot 2^{-\kappa/2} \geq \varepsilon_{A^n}^{\mathscr{Q}} - \sqrt{\frac{n \cdot T^A}{2^\kappa}}. \tag{6.1}$$

Now we set $n = 1/\varepsilon_A^{\mathscr{Q}}$ so that $\varepsilon_{A^n}^{\mathscr{Q}} = 1 - (1 - \varepsilon_A^{\mathscr{Q}})^n > 1 - \exp(-1)$. Substituting into (6.1) and using $\frac{T^A}{\varepsilon_A^{\mathscr{Q}}} < 2^{\kappa-3}$ we get

$$\varepsilon_{A^n}^{\mathscr{P}} > 1 - \exp(-1) - \sqrt{\frac{T^A}{2^\kappa \varepsilon_A^{\mathscr{Q}}}} > 1 - \exp(-1) - 2^{-3/2} \approx 0.279.$$

Finally, to achieve a contradiction, we derive a simple upper bound. By union bound $\varepsilon_{A^n}^{\mathscr{P}} \leq n\varepsilon_A^{\mathscr{P}}$. Since $S^{\mathscr{P}}$ is κ-bit secure, $\varepsilon_A^{\mathscr{P}} \leq T^A/2^\kappa$, which shows that

$$\varepsilon_{A^n}^{\mathscr{P}} \leq \frac{nT^A}{2^\kappa} = \frac{T^A}{2^\kappa \varepsilon_A^{\mathscr{Q}}} < 2^{-3} = 0.125$$

which is smaller than the lower bound. □

This shows that $\delta(\mathscr{P}_\theta, \mathscr{Q}_\theta) \sim 2^{-\kappa/2}$ is sufficient to maintain κ bits of security. This type of analysis was first used in [65][1] for the special case of fixed distributions (i.e. θ is fixed and cannot be chosen by the adversary) and the KL-divergence $\delta = \sqrt{\delta_{\mathrm{KL}}}$, which is efficient (see e.g. [5, 65] for proofs). Lemma 1, in combination with Lemma 15, shows that it is sufficient for algorithms to approximate the probabilities of the target distribution with floating point numbers of precision about half the security parameter. Interestingly, in this setting, it is important to approximate probabilities in floating point, as $\kappa/2$ bits of fixed-point precision is not secure. (See [56] for an attack.)

[1] We remark that the proof in [65] was flawed since it assumes repeatability of unforgeability games.

Approximate Samplers and Decision Primitives

Note that above analysis only applies to search primitives. In this section we extend it to decision games. The next theorem shows that it suffices to approximate a distribution $\mathscr{P}$ up to distance $\delta(\mathscr{P},\mathscr{Q}) \leq 2^{-\kappa/2}$ for an efficient divergence δ in order to maintain almost κ bits of security.

Theorem 11 *Let $S^{\mathscr{P}}$ be a* 1*-bit secrecy game with black-box access to a probability ensemble $(\mathscr{P}_\theta)_\theta$, and δ be a λ-*efficient *measure for any $\lambda \leq \frac{1}{4}$. If $S^{\mathscr{P}}$ is κ-bit secure and $\delta(\mathscr{P}_\theta,\mathscr{Q}_\theta) \leq 2^{\kappa/2}$, then $S^{\mathscr{Q}}$ is $(\kappa-8)$-bit secure.*

The remainder of this section is devoted to proving Theorem 11. We will make use of the results in the previous section and first note that the proof of Lemma 15 actually shows something slightly stronger.

Lemma 16 (variant of Lemma 15) *Let $S^{\mathscr{P}}$ be any security game with black-box access to a probability distribution ensemble $\mathscr{P}_\theta$. For any adversary A with resources T that plays $S^{\mathscr{P}}$ and event E over its output, denote $\gamma_{\mathscr{P}} = \Pr[A \in E]$. For the same event, denote by $\gamma_{\mathscr{Q}}$ the probability of E when A is playing $S^{\mathscr{Q}}$. If $\frac{T}{\gamma_{\mathscr{P}}} \geq 2^k$ and $\delta(\mathscr{P}_\theta,\mathscr{Q}_\theta) \leq 2^{-k/2}$ for any $2^{-k/2}$-efficient δ, then $\frac{T}{\gamma_{\mathscr{Q}}} \geq 2^{k-3}$.*

From Lemma 16 we can derive a bound on the output probability of an adversary when switching the distribution of the scheme.

Corollary 6 *For any adversary A with resources T attacking $S^{\mathscr{P}}$ and any event E over A's output, denote the probability of E by $\gamma_{\mathscr{P}}$. Denote the probability of E over A's output when attacking $S^{\mathscr{Q}}$ by $\gamma_{\mathscr{Q}}$. If δ is $\sqrt{\gamma_{\mathscr{Q}}/16T}$-efficient and $\delta(\mathscr{P}_\theta,\mathscr{Q}_\theta) \leq \sqrt{\gamma_{\mathscr{Q}}/16T}$, then $16\gamma_{\mathscr{P}} \geq \gamma_{\mathscr{Q}}$.*

Proof We use Lemma 15 and set k such that $2^{k-4} = \frac{T}{\gamma_{\mathscr{Q}}}$. This implies that $\frac{T}{\gamma_{\mathscr{Q}}} \geq 2^{k-3}$ is false. Assuming towards a contradiction that $16\gamma_{\mathscr{P}} < \gamma_{\mathscr{Q}}$, we see that

$$2^{k-4} = \frac{T}{\gamma_{\mathscr{Q}}} \leq \frac{T}{16\gamma_{\mathscr{P}}}$$

contradicting Lemma 15. □

With this bound in place, we are ready for the main proof.

Proof[of Theorem 11] Fix any T^A-bounded adversary A against $S^{\mathscr{P}}$, output probability $\alpha^A_{\mathscr{P}}$ and conditional success probability $\beta^A_{\mathscr{P}}$. By assumption we have $\alpha^A_{\mathscr{P}}(2\beta^A_{\mathscr{P}} - 1)^2 \leq T^A/2^{\kappa}$. Denote the output and conditional success probability of A against $S^{\mathscr{Q}}$ by $\alpha^A_{\mathscr{Q}}$ and $\beta^A_{\mathscr{Q}}$. Assume towards contradiction that $\alpha^A_{\mathscr{Q}}(2\beta^A_{\mathscr{Q}} - 1)^2 > T^A/2^{\kappa-8}$.

First we apply Corollary 6 to obtain $\alpha^A_{\mathscr{P}} \geq 2^{-4}\alpha^A_{\mathscr{Q}}$. Note that by assumption $\sqrt{\alpha^A_{\mathscr{Q}}/16T} > 2^{(-\kappa+4)/2} > 2^{-\kappa/2} \geq \delta(\mathscr{P}_\theta, \mathscr{Q}_\theta)$ and that trivially $\sqrt{\alpha^A_{\mathscr{Q}}/16T} \leq \frac{1}{4}$.

We now consider the hypothetical modified games $\hat{S}^{\mathscr{P}}$ and $\hat{S}^{\mathscr{Q}}$, which are the same as $S^{\mathscr{P}}$ and $S^{\mathscr{Q}}$ with the only difference that the adversary has the ability to restart the game with fresh randomness at any time. Consider the adversary B against $\hat{S}$ that simply runs A until $A \neq \perp$ (restarting the game if $A = \perp$) and outputs whatever A returns. Let $\alpha = \min(\alpha^A_{\mathscr{P}}, \alpha^A_{\mathscr{Q}})$ and note that B's resources are $T^B < T^A/\alpha$, its output probability is 1 and the (conditional) success probability is $\beta^B_{\mathscr{P}} = \beta^A_{\mathscr{P}}$ (or $\beta^B_{\mathscr{Q}} = \beta^A_{\mathscr{Q}}$) if playing $\hat{S}^{\mathscr{P}}$ (or $\hat{S}^{\mathscr{Q}}$, respectively).

By the properties of δ and Δ_{SD}, we have $\beta^B_{\mathscr{P}} \geq \beta^B_{\mathscr{Q}} - \sqrt{T^B}\delta(\mathscr{P}_\theta, \mathscr{Q}_\theta)$ and so $2\beta^B_{\mathscr{P}} - 1 \geq 2\beta^B_{\mathscr{Q}} - 1 - 2\sqrt{T^B/2^{\kappa}}$. By assumption we also have that $2\beta^A_{\mathscr{P}} - 1 \leq$

$\sqrt{T^A/\alpha^A_{\mathscr{P}} 2^\kappa}$, which yields

$$\sqrt{\frac{T^A}{\alpha 2^\kappa}} \geq \sqrt{\frac{T^A}{\alpha^A_{\mathscr{P}} 2^\kappa}} \geq 2\beta^B_{\mathscr{Q}} - 1 - 2\sqrt{\frac{T^A}{\alpha 2^\kappa}}$$

because $\beta^B_{\mathscr{P}} = \beta^A_{\mathscr{P}}$, and so

$$2\beta^A_{\mathscr{Q}} - 1 = 2\beta^B_{\mathscr{Q}} - 1 \leq 3\sqrt{\frac{T^A}{\alpha 2^\kappa}}.$$

If $\alpha^A_{\mathscr{Q}} \leq \alpha^A_{\mathscr{P}}$, then $\alpha = \alpha^A_{\mathscr{Q}}$ and the above inequality immediately yields the contradiction. Otherwise, we can derive an upper bound on $\alpha^A_{\mathscr{P}}$ from it:

$$\alpha^A_{\mathscr{P}} \leq \frac{9T^A}{2^\kappa (2\beta^A_{\mathscr{Q}} - 1)^2} < \frac{\alpha^A_{\mathscr{Q}}}{2^4}$$

where the latter inequality follows from the assumption. This contradicts our lower bound above. □

Approximate Convolution

In Chapter 7, we will make use of the Theorem 1 and 2 to reduce the task of generating a specific discrete Gaussian, to generating samples from different distributions. Observe that these theorems assume access to exact samplers. In order to analyze our algorithms, we need to bound the divergence from the true distribution when applying the theorems to samples from a distribution close to the exact Gaussian distributions.

Lemma 17 *Let* Δ *be a useful or efficient* metric. *Let* $A^{\mathscr{P}}$ *be an algorithm querying a distribution ensemble* $\mathscr{P}_\theta$ *at most* q *times. Then we have*

$$\Delta(A^{\mathscr{Q}}, \mathscr{R}) \leq \Delta(A^{\mathscr{P}}, \mathscr{R}) + q \cdot \Delta(\mathscr{P}_\theta, \mathscr{Q}_\theta)$$

for any distribution $\mathscr{R}$ and any ensemble $\mathscr{Q}_\theta$.

Proof By triangle inequality, $\Delta(A^{\mathscr{Q}},\mathscr{R}) \leq \Delta(A^{\mathscr{P}},\mathscr{R}) + \Delta(A^{\mathscr{P}},A^{\mathscr{Q}})$. Let $(\theta_i)_i$ (resp. $(\tilde{\theta}_i)_i$) be the sequence of queries made by $A^{\mathscr{P}}$ (resp. $A^{\mathscr{Q}}$). By data processing inequality $\Delta(A^{\mathscr{P}},A^{\mathscr{Q}}) \leq \Delta((\theta_i,\mathscr{P}_i)_i,(\tilde{\theta}_i,\mathscr{Q}_i)_i)$. The rest follows from sub-additivity and Lemma 14. □

By letting A be the algorithm that performs the convolution as in Theorem 1 and applying Lemma 17 to it with $\mathscr{P}_i = \mathscr{D}_{\Lambda,\mathbf{c}_i,s_i}$ and approximate distributions $\mathscr{Q}_i = \tilde{\mathscr{D}}_{\Lambda,\mathbf{c}_i,s_i}$, we can show that convolving approximate discrete Gaussians results in good approximations of the expected discrete Gaussian. Furthermore, we can also apply Lemma 17 to Theorem 2, if we have a bound on the approximation of the second sampler for *any* center c_2.

As an example, consider again the statistical distance Δ_{SD}. By applying Lemma 17 to the convolutions in Theorem 1 (resp. 2), the resulting approximation error satisfies:

$$\Delta_{\mathrm{SD}}(A^{\tilde{\mathscr{D}}_{\Lambda,\mathbf{c}_i,s_i}},\mathscr{D}_{Y,s}) \lesssim 2\varepsilon + \sum_i \Delta_{\mathrm{SD}}(\tilde{\mathscr{D}}_{\Lambda,\mathbf{c}_i,s_i},\mathscr{D}_{\Lambda,\mathbf{c}_i,s_i}).$$

Conveniently, this works recursively: if we use the obtained approximate samples as input to another convolution, the loss in statistical distance is simply additive in the number of convolutions we apply. This shows that using a metric to analyze approximation errors is relatively straight-forward.

Unfortunately, Δ_{SD} is not cryptographically efficient and thus requires high precision to guarantee security. While $\sqrt{\delta_{\mathrm{KL}}}$ allows to improve on that, it is not a metric and thus Lemma 17 does not apply. One can still use $\sqrt{\delta_{\mathrm{KL}}}$ to improve on the efficiency by exploiting the metric properties of Δ_{SD}, i.e. one first decomposes the statistical distance of the approximate distribution as in the previous paragraph, and then bounds the individual

parts using property 6.1.1. But as we start working with more complex and recursive algorithms, this method becomes more involved. One needs to be careful to not rely on typical metric properties when analyzing algorithms using $\sqrt{\delta_{\mathrm{KL}}}$, like triangle inequality and symmetry. It would be much more convenient to use a cryptographically useful and efficient *metric* Δ. This would allow to carry out the analysis using only Δ, and directly claim bit security of $2\log\Delta(\mathscr{P}_\theta, \mathscr{Q}_\theta)$ by Lemma 15.

A New Closeness Metric

In this section we introduce a new measure of closeness between probability distributions which combines the ease of use of a metric with the properties of divergences that allow to obtain sharper security bounds. More specifically, we provide an efficient metric with a simple definition.

Definition 33 *The* max-log *distance between two distributions $\mathscr{P}$ and $\mathscr{Q}$ over the same support S is*

$$\Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q}) \quad = \quad \max_{x \in S} \left| \ln \mathscr{P}(x) - \ln \mathscr{Q}(x) \right| .$$

For convenience, we also write $\Delta_{\mathrm{ML}}(p, q) = |\ln p - \ln q|$ for any two positive reals p and q. It is easy to see that Δ_{ML} is a metric.

Lemma 18 *Δ_{ML} is a metric, i.e., it is symmetric ($\Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q}) = \Delta_{\mathrm{ML}}(\mathscr{Q}, \mathscr{P})$), positive definite ($\Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q}) \geq 0$ with equality if and only if $\mathscr{P} = \mathscr{Q}$), and it satisfies the triangle inequality ($\Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q}) \leq \Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{R}) + \Delta_{\mathrm{ML}}(\mathscr{R}, \mathscr{Q})$).*

Proof All properties are inherited from the infinity norm, simply by noticing that $\Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q}) = \|f(\mathscr{P}) - f(\mathscr{Q})\|_\infty$ for some function $f(\mathscr{P}) = (\ln \mathscr{P}(x))_x$. □

We note that in the regime close to 0, Δ_{ML} is essentially equal to δ_{RE}.

Lemma 19 *For any two positive real p and q,*

$$\begin{aligned} \Delta_{\mathrm{ML}}(p,q) &\leq -\ln(1-\delta_{\mathrm{RE}}(p,q)) \lesssim \delta_{\mathrm{RE}}(p,q) && (6.2)\\ \delta_{\mathrm{RE}}(p,q) &\leq \exp(\Delta_{\mathrm{ML}}(p,q))-1 \lesssim \Delta_{\mathrm{ML}}(p,q). && (6.3) \end{aligned}$$

The same bounds hold for $\Delta_{\mathrm{ML}}(\mathscr{P},\mathscr{Q})$ and $\delta_{\mathrm{RE}}(\mathscr{P},\mathscr{Q})$ for any two distributions $\mathscr{P},\mathscr{Q}$ over the same support S.

Proof Let $\varepsilon = \delta_{\mathrm{RE}}(p,q)$, so that $(q/p) \in (1 \pm \varepsilon)$. It follows that $\Delta_{\mathrm{ML}}(p,q) = |\ln(p/q)| \leq \max[\ln(1+\varepsilon), \ln(1/(1-\varepsilon))] = -\ln(1-\varepsilon)$. This proves (6.2). For (6.3), let $\varepsilon = \Delta_{\mathrm{ML}}(p,q)$, so that $\max(p/q, q/p) \leq \exp(\varepsilon)$. If $p < q$, then $\delta_{\mathrm{RE}}(p,q) = (q/p) - 1 \leq \exp(\varepsilon) - 1$. Else, $p \geq q$, and $\delta_{\mathrm{RE}}(p,q) = 1 - (q/p) \leq (p/q) - 1 \leq \exp(\varepsilon) - 1$. The same bounds for distributions easily follow by taking the maximum of $\Delta_{\mathrm{ML}}(\mathscr{P}(x), \mathscr{Q}(x))$ and $\delta_{\mathrm{RE}}(\mathscr{P}(x), \mathscr{Q}(x))$ when x ranges over the support of the two distributions. □

The next two lemmas prove that Δ_{ML} is an efficient metric.

Lemma 20 Δ_{ML} *satisfies the sub-additivity property (for joint distributions) and data processing inequality.*

Proof We start by proving the subadditivity property for sequences of length two. The general case follows by induction. By triangle inequality,

$$\begin{aligned} \Delta_{\mathrm{ML}}((X_1,X_2),(Y_1,Y_2)) &\leq \Delta_{\mathrm{ML}}((X_1,X_2),(X_1,[Y_2 \mid Y_1 = X_1])) \\ &+ \Delta_{\mathrm{ML}}((X_1,[Y_2 \mid Y_1 = X_1]),(Y_1,Y_2)) \end{aligned}$$

where $(X_1,[Y_2 \mid Y_1 = X_1])$ is the distribution that selects a pair (x,y) by first choosing x with probability $\Pr\{X_1 = x\}$, and then y with probability $\Pr\{Y_2 = y \mid Y_1 = x\}$. By

definition,

$$\begin{aligned}\Delta_{\mathrm{ML}}((X_1,[Y_2 \mid Y_1 = X_1]),(Y_1,Y_2)) &= \max_{(x,y)} \left| \ln \frac{\Pr\{X_1 = x\} \cdot \Pr\{Y_2 = y \mid Y_1 = x\}}{\Pr\{Y_1 = x\} \cdot \Pr\{Y_2 = y \mid Y_1 = x\}} \right| \\ &= \Delta_{\mathrm{ML}}(X_1, Y_1).\end{aligned}$$

and also

$$\begin{aligned}\Delta_{\mathrm{ML}}((X_1,X_2),(X_1,[Y_2 \mid Y_1 = X_1])) &= \max_{(x,y)} \left| \ln \frac{\Pr\{X_1 = x\} \cdot \Pr\{X_2 = y \mid X_1 = x\}}{\Pr\{X_1 = x\} \cdot \Pr\{Y_2 = y \mid Y_1 = x\}} \right| \\ &= \max_x \Delta_{\mathrm{ML}}([X_2 \mid X_1 = x],[Y_2 \mid Y_1 = x])\end{aligned}$$

This proves subadditivity for joint distributions.

For the data processing inequality, let $\mathscr{P}$ and $\mathscr{Q}$ be two probability distributions with support S and $f\colon S \mapsto T$ be any (deterministic) function. Then

$$\begin{aligned}\Delta_{\mathrm{ML}}(f(\mathscr{P}), f(\mathscr{Q})) &= \max_{j \in T} |\ln \Pr[f(\mathscr{P}) = j] - \ln Pr[f(\mathscr{Q}) = j]| \\ &= \max_{j \in T} \ln \left(\max \left\{ \frac{\sum_{i \in f^{-1}(j)} \mathscr{P}(i)}{\sum_{i \in f^{-1}(j)} \mathscr{Q}(i)}, \frac{\sum_{i \in f^{-1}(j)} \mathscr{Q}(i)}{\sum_{i \in f^{-1}(j)} \mathscr{P}(i)} \right\} \right) \\ &\leq \max_{j \in T} \ln \left(\max \left\{ \max_{i \in f^{-1}(j)} \frac{\mathscr{P}(i)}{\mathscr{Q}(i)}, \max_{i \in f^{-1}(j)} \frac{\mathscr{Q}(i)}{\mathscr{P}(i)} \right\} \right) \\ &= \max_{i \in S} \ln \left(\max \left\{ \frac{\mathscr{P}(i)}{\mathscr{Q}(i)}, \frac{\mathscr{Q}(i)}{\mathscr{P}(i)} \right\} \right) = \Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q})\end{aligned}$$

where the inequality holds, because $(\sum_i a_i)/(\sum_i b_i) \leq \max_i(a_i/b_i)$ for all $b_i \geq 0$. The result for randomized functions follows by treating the random coins as explicit input and combining the above with the sub-additivity property. □

Finally, we show that Δ_{ML} also satisfies the pythagorean probability preservation property for any parameter $\lambda \leq \frac{1}{3}$.

Lemma 21 *For distributions $\mathscr{P}_i$ and $\mathscr{Q}_i$ over support $\prod_i S_i$, if $\Delta_{\mathrm{ML}}(\mathscr{P}_i \mid a_i, \mathscr{Q}_i \mid a_i) \leq 1/3$ for all i and $a_i \in \prod_{j<i} S_j$, then*

$$\Delta_{\mathrm{SD}}((\mathscr{P}_i)_i, (\mathscr{Q}_i)_i) \leq \|(\max_{a_i} \Delta_{\mathrm{ML}}(\mathscr{P}_i \mid a_i, \mathscr{Q}_i \mid a_i))_i\|_2.$$

Proof First, we observe that under the condition $\Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q}) \leq 1/3$, we have

$$\delta_{\mathrm{KL}}(\mathscr{P}, \mathscr{Q}) \leq 2\Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q})^2.$$

This can be checked using Equation (6.3) as follows. Let $x = \Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q}) \leq 1/3$. Applying Lemma 1 with $\mu = e^x - 1$, we get

$$\delta_{\mathrm{KL}}(\mathscr{P}, \mathscr{Q}) \leq \frac{(e^x - 1)^2}{2(2 - e^x)^2} \leq 2x^2,$$

where the last inequality is implied by $(e^x - 1)(1 + 1/(2x)) \leq 1$, which can be verified using the convexity bound $e^x - 1 \leq (e^{\frac{1}{3}} - 1)3x$ (valid for $x \in [0, 1/3]$) as follows:

$$(e^x - 1) \cdot \left(1 + \frac{1}{2x}\right) \leq (e^{\frac{1}{3}} - 1) \cdot (3x + 1.5) \leq (e^{\frac{1}{3}} - 1) \cdot 2.5 \approx 0.99.$$

Now that we have established the bound $\delta_{\mathrm{KL}}(\mathscr{P}, \mathscr{Q}) \leq 2\Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q})^2$, we can use Pinsker's inequality and the sub-additivity of δ_{KL} (which directly follows from what

is often referred to as the *chain rule*) to get

$$\begin{aligned}\Delta_{\mathrm{SD}}((\mathscr{P}_i)_i,(\mathscr{Q}_i)_i) &\leq \sqrt{\delta_{\mathrm{KL}}((\mathscr{P}_i)_i,(\mathscr{Q}_i)_i)/2} \\ &\leq \sqrt{\frac{1}{2}\sum_i \max_{a_i} \delta_{\mathrm{KL}}(\mathscr{P}_i \mid a_i, \mathscr{Q}_i \mid a_i)} \\ &\leq \sqrt{\sum_i \max_{a_i} \Delta_{\mathrm{ML}}(\mathscr{P}_i \mid a_i, \mathscr{Q}_i \mid a_i)^2} \\ &= \|(\max_{a_i} \Delta_{\mathrm{ML}}(\mathscr{P}_i \mid a_i, \mathscr{Q}_i \mid a_i))_i\|_2.\end{aligned}$$

□

It follows that we can instantiate Lemma 17 with Δ_{ML} to analyze the increase of approximation error if applying multiple convolutions to approximate samples. We will make intensive use of this in Chapter 7 to analyze the approximation error of our new sampling algorithm.

Relationship to Other Measures. The max-log distance is closely related to the Rényi divergence of order ∞ and shares many of its properties, including a multiplicative probability preservation: $Pr_{X\leftarrow\mathscr{P}}[E] \geq Pr_{X\leftarrow\mathscr{Q}}[E]/\exp(\Delta_{\mathrm{ML}}(\mathscr{P},\mathscr{Q}))$ [5]. This can be useful for other definitions of bit security than Definition 28. In particular, consider a setting where the number of queries to the distribution is bounded by some fixed number q. This seems reasonable in many applications since queries often require interaction with an honest user. In this setting, it is easy to show that the success probability of an adversary can only increase by a multiplicative factor of $\exp(q\Delta_{\mathrm{ML}}(\mathscr{P}_\theta,\mathscr{Q}_\theta))$ using sub-additivity, data processing inequality, and multiplicative probability preservation. This shows that as long as $\Delta_{\mathrm{ML}}(\mathscr{P},\mathscr{Q}) < 1/q$, one loses almost no security. In a subsequent work [66], Prest shows that by generalizing Lemma 1 to Rényi divergences of

arbitrary order, one can achieve tighter bounds ($\Delta_{\mathrm{ML}}(\mathscr{P},\mathscr{Q}) \approx \sqrt{1/q}$) in this setting by optimizing over the order of the divergence.

It has also been noted that the Rényi divergence is related to the notion of differential privacy. More specifically, an algorithm $A(D)$, taking a database D as input, is ε-differentially private if the Rényi divergence of order ∞ between the output distributions of $A(D_1)$ and $A(D_2)$ is less than ε for any two neighboring databases D_1 and D_2. Since *neighborhood* is often defined using a symmetric relation on the set of databases, this is equivalent to a formulation using the max-log distance. Finally, the techniques used in [66] are related to *advanced composition theorems* in the differential privacy terminology. For more details we refer the reader to [57] and references therein.

Chapter 6, in full, is a combination of

- material as it appears (with minor modifications) in "Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time" by Daniele Micciancio and Michael Walter, published in the proceedings of the Thirty-Seventh Annual International Cryptology Conference (CRYPTO 2017). The dissertation author was the primary investigator and author of this paper.

- material (with minor modifications) that has been submitted for publication and may appear as "On the Bit Security of Cryptographic Primitives" by Daniele Micciancio and Michael Walter. The dissertation author was the primary investigator and author of this paper.

Chapter 7

Gaussian Sampling over the Integers

In this chapter we describe and analyze a new algorithm to sample the discrete Gaussian distribution. The entire algorithm SAMPLEZ is presented in Algorithm 7.1. In Sect. 7.1 and 7.2, we analyze the sub-routines SAMPLEI and SAMPLEC, which may already be directly useful in some applications. Then, in Sect. 7.3, we analyze the full algorithm SAMPLEZ. All algorithms assume access to a base sampler SAMPLEB to approximate the distribution $\mathscr{D}_{c_i+\mathbb{Z},s_0}$, for a small and fixed set of values for the coset c_i and one fixed s_0. Any algorithm can be used as a base sampler, provided it produces distributions $\tilde{\mathscr{D}}_{c_i+\mathbb{Z},s_0}$ within a small distance $\Delta_{\mathrm{ML}}(\tilde{\mathscr{D}}_{c_i+\mathbb{Z},s_0}, \mathscr{D}_{c_i+\mathbb{Z},s_0}) \leq \mu$ from the exact Gaussian $\mathscr{D}_{c_i+\mathbb{Z},s_0}$. By Lemma 19, this is essentially equivalent to approximating the Gaussian probabilities with a relative error bound of μ. The reader is referred to Sect. 7.5.2 for a possible choice of SAMPLEB.

Before we continue, we define rounding operators $\lceil c \rceil_k = \lceil 2^k c \rceil / 2^k$ and $\lfloor c \rfloor_k = \lfloor 2^k c \rfloor / 2^k$ for $c \in [0,1)$ and $k \in \mathbb{Z}$, which round c (up or down, respectively) to a number with k fractional bits. We also define a randomized rounding operator $\lfloor c \rceil_k = \lfloor c \rfloor_k + \mathscr{B}_\alpha / 2^k$ (where $\mathscr{B}_\alpha$ is a Bernoulli random variable of parameter $\alpha = 2^k c \mod 1$) which rounds c to either $\lceil c \rceil_k$ (with probability α) or $\lfloor c \rfloor_k$ (with probability $1-\alpha$).

Finally, we saw in Section 6.2 that we can instantiate Lemma 17 with Δ_{ML} to analyze the increase of approximation error if applying multiple convolutions to

approximate samples. For convenience, we reformulate Theorem 1 and 2 in terms of the max-log distance and approximate distributions (following Lemma 17), specializing them to our setting.

Corollary 7 *Let $\mathbf{z} \in \mathbb{Z}^m$ be a nonzero integer vector with $gcd(\mathbf{z}) = 1$ and $s \in \mathbb{R}^m$ with $s_i \geq \sqrt{2}\|\mathbf{z}\|_\infty \eta_\varepsilon(\mathbb{Z})$ for all $i \leq m$. Let y_i be independent samples from $\tilde{\mathscr{D}}_{\mathbb{Z},s_i}$, respectively, with $\Delta_{\mathrm{ML}}(\mathscr{D}_{\mathbb{Z},s_i}, \tilde{\mathscr{D}}_{\mathbb{Z},s_i}) \leq \mu_i$ for all i. Let $\tilde{\mathscr{D}}_{\mathbb{Z},s}$ be the distribution of $y = \sum z_i y_i$. Then $\Delta_{\mathrm{ML}}(\mathscr{D}_{\mathbb{Z},s}, \tilde{\mathscr{D}}_{\mathbb{Z},s}) \lesssim 2\varepsilon + \sum_i \mu_i$.*

Corollary 8 *Let $s_1, s_2 > 0$, with $s^2 = s_1^2 + s_2^2$ and $s_3^{-2} = s_1^{-2} + s_2^{-2}$. Let $\Lambda = K\mathbb{Z}$ be a copy of the integer lattice $\mathbb{Z}$ scaled by a constant K. For any c_1 and $c_2 \in \mathbb{R}$, denote the distribution of $x_1 \leftarrow x_2 + \tilde{\mathscr{D}}_{c_1 - x_2 + \mathbb{Z}, s_1}$, where $x_2 \leftarrow \tilde{\mathscr{D}}_{c_2+\Lambda,s_2}$, by $\tilde{\mathscr{D}}_{c_1+\mathbb{Z},s}$. If $s_1 \geq \eta_\varepsilon(\mathbb{Z})$, $s_3 \geq \eta_\varepsilon(\Lambda) = K\eta_\varepsilon(\mathbb{Z})$, $\Delta_{\mathrm{ML}}(\mathscr{D}_{c_2+\Lambda,s_2}, \tilde{\mathscr{D}}_{c_2+\Lambda,s_2}) \leq \mu_2$ and $\Delta_{\mathrm{ML}}(\mathscr{D}_{c+\mathbb{Z},s_1}, \tilde{\mathscr{D}}_{c+\mathbb{Z},s_1}) \leq \mu_1$ for any $c \in \mathbb{R}$, then $\Delta_{\mathrm{ML}}(\mathscr{D}_{c_1+\mathbb{Z},s}, \tilde{\mathscr{D}}_{c_1+\mathbb{Z},s}) \lesssim 4\varepsilon + \mu_1 + \mu_2$.*

Algorithm 7.1. A sampling algorithm for $\mathscr{D}_{c+\mathbb{Z},s}$ for arbitrary c and s. Definitions for z_i and s_i as in (7.1) and (7.2) and $\bar{s}$ as in (7.3). SAMPLEB is an arbitrary base sampler for $\mathscr{D}_{c+\mathbb{Z},s_0}$ with fixed s_0 and small number of cosets $c+\mathbb{Z}$, where $c \in \mathbb{Z}/b$.

SAMPLEZ$_{b,k,\max}(c,s)$

$x \leftarrow$ SAMPLEI(max)

$K \leftarrow \sqrt{s^2-\bar{s}^2}/s_{\max}$

$c' \leftarrow \lfloor c+Kx \rceil_k$

$y \leftarrow$ SAMPLEC$_{b,s_0}(c')$

return y

SAMPLEC$_b(c \in b^{-k}\mathbb{Z})$

if $k=0$

return 0

$g \leftarrow b^{-k+1} \cdot$ SAMPLEB$_{s_0}(b^{k-1}c)$

return $g +$ SAMPLEC$_b(c-g \in b^{-k+1}\mathbb{Z})$

SAMPLEI(i)

if $i=0$

$x \leftarrow$ SAMPLEB$_{s_0}(0)$

return x

$x_1 \leftarrow$ SAMPLEI$(i-1)$

$x_2 \leftarrow$ SAMPLEI$(i-1)$

$y = z_i x_1 + \max(1, z_i - 1)x_2$

return y

Large deviations

In this section we show how to efficiently sample $\mathscr{D}_{\mathbb{Z},s}$ for an arbitrarily large $s \gg \eta_\varepsilon(\mathbb{Z})$ using samples from $\mathscr{D}_{\mathbb{Z},s_0}$ for some small fixed value of $s_0 \geq \sqrt{2}\eta_\varepsilon(\mathbb{Z})$. For this we make use of convolution to combine the samples from the basic sampler to yield a distribution with larger noise parameter. The algorithm accomplishing this is given in Algorithm 7.1 as SAMPLEI.

Lemma 22 *For a given value of $s_0 \geq 4\sqrt{2}\eta_\varepsilon(\mathbb{Z})$ define the following sequence of values*[1]

[1]Notice that the values in (7.1) and (7.2) depend both on the index i and the initial s_0, so we will write

for $i > 0$:

$$z_i = \left\lfloor \frac{s_{i-1}}{\sqrt{2}\eta} \right\rfloor \tag{7.1}$$

$$s_i^2 = (z_i^2 + \max((z_i - 1)^2, 1))s_{i-1}^2 \tag{7.2}$$

If $\Delta_{\mathrm{ML}}(\mathscr{D}_{\mathbb{Z},s_0}, \textsc{SampleB}_{s_0}(0)) \leq \mu$, then $\Delta_{\mathrm{ML}}(\mathscr{D}_{\mathbb{Z},s_i}, \textsc{SampleI}(i)) \leq (\mu + 2\varepsilon)2^i$ and the running time of SAMPLEI *is at most 2^i plus 2^i invocations of* SAMPLEB. *Finally, $s_i(s_0) \geq 2^{2^i}$, implying $i \leq \lceil \log\log s \rceil$ is sufficient to achieve a given target s.*

Proof Note that SAMPLEI repeatedly invokes Corollary 7. The conditions of Corollary 7 are met by definition of z_i (Equation (7.1)), so every application incurs a loss in Δ_{ML} of 2ε by Corollary 7. The bound on the number of base samples and convolutions is immediate.

We conclude by proving the statement $s_i(s_0) \geq 2^{2^i}$ under the conditions of the lemma. Let $\eta = \eta_\varepsilon(\mathbb{Z})$. By definition we have $z_i \geq \frac{s_{i-1}}{\sqrt{2}\eta} - 1$ and so

$$s_i^2 \geq 2s_{i-1}^2 \left(\frac{s_{i-1}}{\sqrt{2}\eta} - 2 \right)^2$$

and so

$$s_i \geq \sqrt{2}s_{i-1} \left(\frac{s_{i-1}}{\sqrt{2}\eta} - 2 \right) \geq s_{i-1}^2 \left(\frac{1}{\eta} - \frac{2\sqrt{2}}{s_{i-1}} \right) \geq s_{i-1}^2 \left(\frac{1}{\eta} - \frac{2\sqrt{2}}{s_0} \right) \geq \frac{s_{i-1}^2}{2\eta}.$$

Equivalently,

$$\log s_i \geq 2\log s_{i-1} - \log 2\eta.$$

---

them as $z_i(s_0)$ and $s_i(s_0)$ when we need to emphasize this dependency.

Unrolling the recursion, we obtain

$$\log s_i \geq 2^i \log s_0 - \left(\sum_{j=0}^{i-1} 2^j\right) \log 2\eta = 2^i \log s_0 - (2^i - 1) \log 2\eta \geq 2^i (\log s_0 - \log 2\eta) \geq 2^i$$

and so $s_i \geq 2^{2^i}$. $\square$

The algorithm SAMPLEI will overshoot the noise parameter, but in many applications (including ours further below) this is enough. In fact, for us it will not matter by how much we overshoot a given target s, as we will show in the following sections how to adjust the noise parameter to obtain a sample from a specific target distribution (with arbitrary center).

Algorithm 7.2. A sampling algorithm for $\mathscr{D}_{\mathbb{Z},\tilde{s}}$ for some $\tilde{s}$ not much larger than s. Definition of s_i as in (7.2).

SAMPLECENTEREDGAUSSIAN(s)

Select largest i such that $s_i < s$

$x_1 \leftarrow$ SAMPLEI(i)

$x_2 \leftarrow$ SAMPLEI(i)

$z \leftarrow \left\lceil \frac{1}{2}\left(1 + \sqrt{2\left(\frac{s}{s_i}\right)^2 - 1}\right) \right\rceil$

return $zx_1 + (z-1)x_2$

If all we are interested in is the centered Gaussian distribution with a specific noise parameter not much larger than a certain target width, as is the case in many applications, it is relatively easy to adapt the algorithm to get closer to the target s. One way of doing this is to adjust z_i in the top level of the recursion to yield something closer to s. This is demonstrated by Algorithm 7.2, for which the following corollary establishes

a bound on the size of the resulting noise parameter.

Corollary 9 *If* $\Delta_{\mathrm{ML}}(\textsc{SampleI}(i), \mathscr{D}_{\mathbb{Z},s_i}) \lesssim \mu$ *for the largest* i *such that* $s_i \leq s$ *and* $s \geq s_0 \geq \sqrt{2}\eta_\varepsilon(\mathbb{Z})$*, then* $\Delta_{\mathrm{ML}}(\textsc{SampleCenteredGaussian}(s), \mathscr{D}_{\mathbb{Z},\tilde{s}}) \lesssim 2\mu + 2\varepsilon$ *for some* $\tilde{s}$ *such that* $s \leq \tilde{s} \leq \sqrt{5}s$.

Proof First note that $s_i < s$ implies $z \geq 2$. The choice of z and s_i now guarantees that Corollary 7 is applicable and that $(z-1)^2 + (z-2)^2 < \frac{s^2}{s_i^2} \leq z^2 + (z-1)^2$. Since $\tilde{s}^2 = (z^2 + (z-1)^2)s_i^2$ this establishes the lower bound and shows that $\tilde{s}^2 \leq \frac{z^2+(z-1)^2}{(z-1)^2+(z-2)^2} s^2$. The upper bound follows from the fact that the ratio $\frac{z^2+(z-1)^2}{(z-1)^2+(z-2)^2}$ is decreasing in z and equals 5 for $z = 2$.

The bound on the Δ_{ML} distance is immediate from Corollary 7. □

Note that the constant $\sqrt{5}$ in Corollary 9 follows from the worst case where $z = 2$. Using a little more care in the choice of small coefficients, the bound can be improved to $\sqrt{2}$, but for a simpler exposition we omitted this optimization. However, it will not be possible to get arbitrarily close to any target s if given a fixed s_0, but if the target s is fixed we can always choose a suitable small s_0 such that the target distribution will be generated exactly.

For a fixed s_0, $z_i(s_0)$ and $s_i(s_0)$ are fixed, so one can precompute s_i and corresponding z_i for a small set of i. As Lemma 22 shows, the s_i grow very rapidly so only a very small number ($\sim \log\log s$) of precomputed values are necessary to generate extremely wide distributions. If the target s is fixed, only the coefficients z_i need to be stored.

Arbitrary center

We now show how to sample from an arbitrary coset $c+\mathbb{Z}$ using samplers for only a small number of cosets. We assume c is given as a k digit number in base b between 0 and 1. The parameter k dictates the trade-off between running time and output precision, while the basis b determines the number of cosets the base sampler SAMPLEB needs to be able to sample from.

The idea of our new algorithm SAMPLEC (see Algorithm 7.1) is to round the center randomly digit by digit to finally obtain a sample from $c+\mathbb{Z}$. Every rounding operation consumes a sample from one of b cosets of $\mathbb{Z}$ (where b is a parameter). To show correctness, we iteratively use a convolution theorem. While this process of iterative rounding increases the noise of the output distribution, this increase is minor as the following lemma shows.

Lemma 23 *Let $2 \leq b \in \mathbb{Z}$ be a base, $s_0 \geq (\sqrt{(b+1)/b})\eta_\varepsilon(\mathbb{Z})$ and $c \in b^{-k}\mathbb{Z}$. If*

$$\Delta_{\mathrm{ML}}(\mathscr{D}_{c_i+\mathbb{Z},s_0}, \textsc{SampleB}_{s_0}(c_i)) \leq \mu$$

for all $c_i \in \mathbb{Z}/b$, then $\Delta_{\mathrm{ML}}(\textsc{SampleC}_b(c), \mathscr{D}_{c+\mathbb{Z},\bar{s}}) \lesssim (4\varepsilon+\mu)k$ where

$$\bar{s} = s_0 \left(\sqrt{\sum_{i=0}^{k-1} b^{-2i}} \right). \tag{7.3}$$

Proof The proof follows by induction and Corollary 8. For $k = 1$ the claim is obviously true. For $k > 1$, invoke the induction hypothesis and apply Corollary 8 with $s_1 = s_0\sqrt{\sum_{i=0}^{k-2} b^{-2i}}$, $s_2 = s_0/b^{k-1}$, $\Lambda = b^{-k+1}\mathbb{Z}$, $c_2 = b^{-k}[c]_k$ (where $[c]_k$ is the k-th digit in the b-ary expansion of c), and $c_1 = c$.

It remains to show that the conditions on the noise parameters are met. First note

that $\sum_{i=0}^{k} b^{-2i} \geq 1$ for all $k \geq 1$, and so $s_1 \geq s_0 > \eta_\varepsilon(\mathbb{Z})$.

Then we have

$$s_3^{-2} = s_1^{-2} + s_2^{-2} = s_0^{-2}\left(\left(\sum_{i=0}^{k-2} b^{-2i}\right)^{-1} + b^{2(k-1)}\right)$$
$$= s_0^{-2}\left(\frac{1-b^{-2}}{1-b^{-2(k-1)}} + b^{2(k-1)}\right) = s_0^{-2}\frac{b^{2(k-1)}-b^{-2}}{1-b^{-2(k-1)}}$$

and so

$$s_3 = \sqrt{\frac{1-b^{-2(k-1)}}{b^{2(k-1)}-b^{-2}}}s_0 = \frac{1}{b^{k-1}}\sqrt{\frac{1-b^{-2(k-1)}}{1-b^{-2k}}}s_0 = \frac{1}{b^{k-1}}\sqrt{\frac{b^{2k}-b^2}{b^{2k}-1}}s_0$$

Note that

$$\frac{b+1}{b}\cdot\frac{b^{2k}-b^2}{b^{2k}-1} \geq 1$$

for all $k > 1$, which shows that $s_3 \geq b^{-k+1}\eta_\varepsilon(\mathbb{Z}) = \eta_\varepsilon(\Lambda)$. □

The parameter b in SAMPLEC offers a trade-off between running time and number of required samplers for cosets of $\mathbb{Z}$. As most efficient samplers require storage for each coset, this is effectively a time-memory trade-off. The larger the base b, the more bits we can round at a time, but that requires more cosets. Note that the running time decreases by a logarithmic factor in b, while the storage requirement increases linearly with b.

Reducing the number of required samples

Recall from the previous section that the parameter k determines the trade-off between running time and output precision: the larger k, the closer the approximation of the centers and thus the better the output distribution, but the number of required base samples and the running time grow linearly with k. We now show that by using a biased coin flip we can speed up the algorithm by a factor 2 while maintaining a good

approximation.

Lemma 24 *Let $s \geq \eta_\varepsilon(\mathbb{Z})$ and $b,k \in \mathbb{Z}$ such that $\tau = b^{-k} \leq (4\pi)^{-1}$. Then*

$$\Delta_{\mathrm{ML}}(\mathscr{D}_{\mathbb{Z},c,s}, \mathscr{D}_{\mathbb{Z},\lfloor c \rceil_k,s}) \lesssim \pi^2\tau^2 + 2\varepsilon = \pi^2/b^{2k} + 2\varepsilon,$$

where $\mathscr{D}_{\mathbb{Z},\lfloor c \rceil_k,s}$ is the distribution of the process of computing $c' = \lfloor c \rceil_k$ and then returning a sample from $\mathscr{D}_{\mathbb{Z},c',s}$.

To prove the lemma, we first observe that linear functions can approximate the Gaussian function well on small enough intervals.

Lemma 25 *For any x_1, x_2 with $x_2 - x_1 = \tau$, $|x_1|, |x_2| \leq ts$ for some $t \geq 1$ and $x \in [x_1, x_2]$, we have*

$$\delta_{\mathrm{RE}}\left(\rho_s(x), \frac{x-x_1}{\tau}\rho_s(x_2) + \frac{x_2-x}{\tau}\rho_s(x_1)\right) \leq \frac{\pi^2 t^2 \tau^2}{2s^2} e^{\frac{2\pi\tau t}{s}}.$$

In particular, if $\tau \leq \frac{s}{4\pi t}$, the bound on the right hand side is less than $\frac{\pi^2 t^2 \tau^2}{s^2}$.

Proof By linear interpolation,

$$\left|\rho_s(x) - \left(\frac{x-x_1}{\tau}\rho_s(x_2) + \frac{x_2-x}{\tau}\rho_s(x_1)\right)\right| \leq \frac{\tau^2}{8} \max_{x_1 \leq x' \leq x_2} |\rho_s''(x')|$$

Observe that

$$\rho_s''(x) = \left(\frac{2\pi x^2}{s^2} - 1\right)\frac{2\pi}{s^2}\rho_s(x)$$

implying that $\|\rho_s''(x')\| \leq \max(\frac{2\pi x'^2}{s^2}, 1)\frac{2\pi}{s^2}\rho_s(x') \leq \frac{4\pi^2 t^2}{s^2}\rho(x')$. Finally note that if $x'^2 \geq x^2$, then $\rho_s(x') \leq \rho_s(x)$. Otherwise,

$$\frac{\rho_s(x')}{\rho_s(x)} = e^{-\pi(\frac{x'^2-x^2}{s^2})} = e^{\pi(\frac{x^2-x'^2}{s^2})} = e^{\pi(\frac{(x-x')(x+x')}{s^2})} \leq e^{\frac{2\pi\tau t}{s}}$$

concluding the proof. □

Proof [of Lemma 24] We set $t = \eta_\varepsilon(\mathbb{Z})$, which allows us to treat $\mathscr{D}_{c+\mathbb{Z},s}$ as a ts-bounded distribution. If we assume that $s \geq \eta_\varepsilon(\mathbb{Z})$ for some negligible ε, we can conclude that Lemma 25 also holds for the respective distributions, since $\rho_s(c+\mathbb{Z}) \approx s$ for any c, i.e. with $c_1 = \lfloor c \rfloor_k$ and $c_2 = \lceil c \rceil_k$:

$$\begin{aligned}
\Delta_{\mathrm{ML}}(\mathscr{D}_{\mathbb{Z},c,s}, \mathscr{D}_{\mathbb{Z},\lfloor c \rceil_r,s}) &= \max_x \left| \ln \frac{\mathscr{D}_{\mathbb{Z},c,s}(x)}{\mathscr{D}_{\mathbb{Z},\lfloor c \rceil_r,s}(x)} \right| \\
&= \max_x \left| \ln \frac{\mathscr{D}_{\mathbb{Z},c,s}(x)}{\left(\frac{c_2-c}{\tau}\mathscr{D}_{\mathbb{Z},c_1,s}(x) + \frac{c-c_1}{\tau}\mathscr{D}_{\mathbb{Z},c_2,s}(x)\right)} \right| \\
&\leq \max_x \left| \ln \frac{\rho_s(x-c)(1\pm\varepsilon)s}{(1\pm\varepsilon)s\left(\frac{c_2-c}{\tau}\rho_s(x-c_1) + \frac{c-c_1}{\tau}\rho_s(x-c_2)\right)} \right| \\
&\leq \max_x \left| \Delta_{\mathrm{ML}}\left(\rho_s(x-c), \frac{c_2-c}{\tau}\rho_s(x-c_1) + \frac{c-c_1}{\tau}\rho_s(x-c_2)\right) + \ln\frac{1\pm\varepsilon}{1\pm\varepsilon} \right| \\
&\lesssim \max_x \delta_{\mathrm{RE}}\left(\rho_s(x-c), \frac{c_2-c}{\tau}\rho_s(x-c_1) + \frac{c-c_1}{\tau}\rho_s(x-c_2)\right) + 2\varepsilon \\
&\leq \frac{\pi^2 t^2 \tau^2}{s^2} + 2\varepsilon \\
&\lesssim \frac{\pi^2}{b^{2k}} + 2\varepsilon
\end{aligned}$$

where we used Lemma 25 and Lemma 19. □

In combination with SAMPLEC (cf. Algorithm 7.1), Lemma 24 suggests an efficient algorithm to sample from $\mathscr{D}_{\mathbb{Z},c,\bar{s}}$ for fixed s and arbitrary c:

1. write c in base b (which is a parameter of the algorithm) and divide this representation into the $k = \log_b \frac{1}{\tau}$ higher order digits (representing c_{head}) and the rest c_{tail}
2. use c_{tail} to define the bias of a Bernoulli distribution to round c_{head} either up or

down

3. return $\text{SAMPLEC}_{b,s_0}(c_{\text{head}} \in b^{-k}\mathbb{Z})$.

These steps correspond to the computation of c' and the following invocation of SAMPLEC in the algorithm SAMPLEZ. The efficiency gain stems from the fact that sampling from a biased Bernoulli distribution is much cheaper than drawing samples from the discrete Gaussian. This allows us to support centers c with arbitrary precision above k with essentially no efficiency loss, since the lower order bits only define the bias of the Bernoulli distribution, which is cheap to implement.

The Full Sampler

So far we have shown how to generate samples efficiently from $\mathscr{D}_{\mathbb{Z},s_i}$ for potentially very large s_i and how to sample from $\mathscr{D}_{\mathbb{Z},c,\bar{s}}$ for arbitrary $c \in \mathbb{R}$ and a specific $\bar{s}$, both using only b samplers for $\mathscr{D}_{\mathbb{Z},c_i,s_0}$ for $c_i \in b^{-1}\mathbb{Z}$ and fixed $s_0 \geq \eta_\varepsilon(\mathbb{Z})$. We now prove correctness of the full sampler, SAMPLEZ, which puts all the pieces together by leveraging Corollary 8 yet again.

Lemma 26 *Let $b,k \in \mathbb{Z}$ be a base and a precision parameter such that $k > \log_b 4\pi$. If*

- $\Delta_{\text{ML}}(\mathscr{D}_{\mathbb{Z},s_{\max}}, \text{SAMPLEI}(\max)) \leq \mu_i$ *and*
- $\Delta_{\text{ML}}(\mathscr{D}_{c'+\mathbb{Z},\bar{s}}, \text{SAMPLEC}_b(c')) \leq \mu_c$ *for any $c' \in \mathbb{Z}/b^k$ and some $\bar{s} \geq \eta_\varepsilon(\mathbb{Z})$,*

then

$$\Delta_{\text{ML}}(\mathscr{D}_{c+\mathbb{Z},s}, \text{SAMPLEZ}_{b,k,\max}(c,s)) \lesssim 6\varepsilon + \pi^2/b^{2k} + \mu_i + \mu_c$$

for any c and s such that $1 < s/\bar{s} \leq s_{\max}/\eta_\varepsilon(\mathbb{Z})$.

Proof By Lemma 24 and 17, $\Delta_{\text{ML}}(\mathscr{D}_{c+Kx}, \text{SAMPLEC}(\lfloor c + Kx \rceil_k)) \leq \pi^2/b^{2k} + 2\varepsilon + \mu_c$. By correctness of SAMPLEI (Lemma 22), $\Delta_{\text{ML}}(\mathscr{D}_{K\mathbb{Z},Ks_{\max}}, Kx) \leq \mu_i$ (where

$x \leftarrow \text{SAMPLEI}(\max)$) and by definition of K we have $s = \sqrt{(Ks_{\max})^2 + \bar{s}^2}$. Now rewrite $\mathscr{D}_{\mathbb{Z},c+Kx,\bar{s}} = c + Kx + \mathscr{D}_{-Kx-c+\mathbb{Z},\bar{s}}$ and apply Corollary 8 with $c_2 = 0$, $c_1 = c$, $x_1 = Kx$ and $x_2 = y$ to see that $\Delta_{\mathrm{ML}}(\mathscr{D}_{c+\mathbb{Z},s}, \text{SAMPLEZ}_{b,k,\max}(c,s)) \lesssim 6\varepsilon + \pi^2/b^{2k} + \mu_i + \mu_c$, if the conditions in the theorem are met. This can easily be seen to be true from the assumptions on s by the following calculation.

$$\begin{aligned} s_3 = \left((Ks_{\max})^{-2} + \bar{s}^{-2}\right)^{-\frac{1}{2}} &= \left(\frac{1}{s^2 - \bar{s}^2} + \frac{1}{\bar{s}^2}\right)^{-\frac{1}{2}} = \left(\frac{\bar{s}^2(s^2 - \bar{s}^2)}{s^2}\right)^{\frac{1}{2}} \\ &= \frac{\bar{s}}{s}\sqrt{s^2 - \bar{s}^2} \geq \sqrt{s^2 - \bar{s}^2}\eta_\varepsilon(\mathbb{Z})/s_{\max} = \eta_\varepsilon(K\mathbb{Z}) \end{aligned}$$

□

The running time of SAMPLEZ is obvious: one invocation of SAMPLEI and one of SAMPLEC, which we analyzed in Sect. 7.1 and 7.2, resp., and a few additional arithmetic operations to calculate K and c'. It is worth noting that the computation of K, the most complex arithmetic computation of the entire algorithm, depends only on s. In many applications, for example trapdoor sampling, s is restricted to a relatively small set, which depends on the key. This means that K_s can be precomputed for the set of possible s's allowing to avoid the FP computation at very low memory cost. Finally, the algorithm may approximate the scaling factor K by a value $\tilde{K}$ such that $\delta_{\mathrm{RE}}(\tilde{K}, K) \leq \mu_K$, which results in an approximation of the distribution of width $\tilde{s} = \sqrt{(\tilde{K}s_i)^2 + \bar{s}}$ instead of s. Elementary calculations show that $\Delta_{\mathrm{ML}}(\mathscr{D}_{\mathbb{Z},c,s}, \mathscr{D}_{\mathbb{Z},c,\tilde{s}}) \lesssim 4\pi t^2 \mu_K$ which by triangle inequality adds to the approximation error.

As an example, assume we have an application, where we know that $\bar{s} \leq s \leq 2^{20} = s_{\max}$. It can be checked, that for any base b and $s_0 \geq 4\sqrt{2}\eta_\varepsilon(\mathbb{Z})$, the following

parameter settings for our algorithm result in

$$\Delta_{\mathrm{ML}}(\mathscr{D}_{\mathbb{Z},c,s}, \textsc{SampleZ}_{b,k,\max}(c,s)) \leq 2^{-52},$$

and thus in ≥ 100 bits of security by Lemma 15:

- $t = \eta_\varepsilon(\mathbb{Z}) = 6$, which results in $\varepsilon \leq 2^{-112}$
- $\mu = 2^{-60}$, the precision of the base sampler, resulting in $\mu_i \leq 2^{-55}$
- $k = \lceil 30/\log b \rceil$, which results in $\mu_c \leq 2^{-55}$ and $\pi^2/b^{2k} \leq 2^{-56}$
- $\mu_K = 2^{-64}$, the precision of calculating K, resulting in $4\pi t^2 \mu_K \leq 2^{-55}$.

Online-Offline Phase and Constant-Time Implementation

Note that a large part of the computation time during our convolution algorithm is spent in the base sampler, which is independent of the center and the noise parameter. This allows us to split the algorithm into an offline and an online phase, similar in spirit to Peikert's sampler [63], which gives rise to a number of platform dependent optimizations. The obvious approach is to simply precompute a number of samples for each of the b cosets and combine them in the online phase until we run out. Note that the trade-off now is not only a time-memory trade-off anymore, it is a time-memory-lifetime trade-off for the device that depends on b. Increasing b speeds up the algorithm, but requires to precompute and store samples for more cosets. While it also means that we effectively decrease the number of samples required per output sample, the latter dependence is only logarithmic, while the former is linear in b.

There are a number of other ways to exploit this structure without limiting the lifetime of the device. Most devices that execute cryptographic primitives have idle times

(e.g. web servers) which can be used to restock the number of precomputed samples. As another example, one can separate the offline phase (basic sampler) and the online phase (combination phase) into two parallel devices with a shared buffer. While the basic sampler keeps filling the buffer with samples, the online phase can combine these samples into the desired distribution. An obvious architecture for such a high performance system would implement the base sampler in a highly parallel fashion (e.g. FPGA or GPU) and the online phase on a regular CPU. This shows that in many scenarios the offline phase can be for free.

The separation of offline and online phase also allows for a straight-forward constant-time implementation with very little overhead. A general problem with sampling algorithms in this context is that the running time of the sampler can leak information about the output sample or the input, which clearly hurts security. For a fixed Gaussian, a simple mitigation strategy is to generate the samples in large batches. This approach breaks down in general when the parameters of the target distribution vary per sample and are not known in advance. In contrast, this idea can be used to implement our algorithm in constant time by generating the basic samples in batches in constant time. Note that every output sample requires the exact same number of base samples and convolutions, so the online phase lends itself naturally to a constant-time implementation.

Assume every invocation of SAMPLEZ requires q base samples and let $\hat{t}_0$ be the maximum over $c_i \in \mathbb{Z}/b$ of the expected running time (over the random coins) of the base sampler (computed either by analysis or experimentation). Consider the following algorithm.

Initialization:

- Use the base sampler to fill b buffers of size q, where the i-th buffer stores discrete Gaussian samples $\mathscr{D}_{c_i+\mathbb{Z},s_o}$ for all $c_i \in \mathbb{Z}/b$.

Query phase:

- On input c and s, call SAMPLEZ(c,s), where SAMPLEB$_{s_0}(c_i)$ simply reads from the respective buffer.
- Call the base sampler q times to restock the buffers and pad the running time of this step to $T = q\hat{t}_0 + O(\sqrt{\kappa q})$.

Note that the restocking of base samples in the query phase runs in constant time with overwhelming probability, which follows from Hoeffding's inequality (the constant in the O-notation depends on the worst-case running time of the base sampler). It follows, that the query phase runs in constant time if all the arithmetic operations in SAMPLEZ are implemented in constant time and the randomized rounding operation is converted to constant time, both of which are easy to achieve.

The amortized overhead is only $O(\sqrt{\kappa/q})$, where q is the number of base samples required per output sample. This can be further reduced, if enough memory for larger buffers is available. Finally, the separation of online and offline phase into different independent systems or precomputation of the offline phase allow for an even more convenient constant-time implementation: One only needs to convert the arithmetic operations and the coin flip into constant time. This incurs only a minimal penalty in running time.

Applications and Comparison

We first give a short overview of existing sampling algorithms (Sect. 7.5.1) and select a suitable one as our base sampler, before we describe the experimental study.

Table 7.1. Comparison of Sampling Algorithms, starting with rejection-based sampler, followed by tree-traversal samplers and finally Algorithm 7.1. The column $\exp(\cdot)$ indicates if the algorithm requires to evaluate $\exp(\cdot)$ online. The column "Generic" refers to the property of being able to produce samples from discrete Gaussians with different parameters not known before precomputation (i.e. which may vary from query to query). The security parameter is denoted by κ.

Algorithm	Memory	Rejection Rate	$\exp(\cdot)$	Generic
Rejection Sampling [28]	0	$\sim .9$	Yes	Yes
Discrete Ziggurat [14]	var	var	Yes	No
Bernoulli-type [19]	$O(\kappa \log s)$	$\sim .5$	No	No
Karney [37]	0	$\sim .5$	No	Yes
Knuth-Yao [21]	$O(\kappa s)$	-	No	No
Inversion Sampling [63]	$O(\kappa s)$	-	No	No
Our work	var	-	No	Yes

Brief Survey of Existing Samplers

All of the currently known samplers can be categorized into two types[2]: rejection-based samplers and tree traversal algorithms. Table 7.1 summarizes the existing sampling algorithms and their properties in comparison to our work. The table does not contain a column with the running time, since this depends on a lot of factors (speed of FP arithmetic vs memory access vs randomness etc.), but for the rejection-based samplers, the rejection rate can be thought of as a measure of the running time. Tree-traversal algorithms should be thought of as much faster than rejection based samplers. A more concrete comparison on a specific platform will be given in Sect. 7.5.4 and Sect. 7.5.6.

The Base Sampler

As mentioned above, our algorithm relies on a base sampler to be available, so we now consider the problem of generating samples from $\mathscr{D}_{\mathbb{Z},c,s}$ when $s = O(\eta_\varepsilon(\mathbb{Z}))$ is relatively small and c is fixed. We are interested in the amortized cost of sample generation, where we want to generate a large batch of samples.

[2]Technically, even rejection-based samplers can be thought of as tree traversal algorithms, but this is not as natural for them, hence our categorization.

We first observe that we are sampling from a relatively narrow Gaussian distribution, so memory will not be a concern for us. For example, assume we choose $s \approx 34 > 4\sqrt{2}\eta_\varepsilon(\mathbb{Z})$ for reasonable ε, and the tailbound parameter $t = 6$ and store all probabilities, i.e. $\mathscr{D}_{c+\mathbb{Z},s}(i)$ for all $0 \leq i \leq ts$ with $i \in c+\mathbb{Z}$, with 64 bit precision. Then we obtain a memory requirement of only $\sim$ 1.5kb for each of the b cosets. Note that storing half the probability table is sufficient in this case, which is obvious if $c \in \{0, 1/2\}$, but is also true for other c since we can exploit symmetries in the different tables that we store. If indeed less memory is available, one can reduce $s \geq \sqrt{2}\eta(\mathbb{Z})$, which is the minimum to be usable for our algorithms. This will come at a moderate cost in performance.

Finally, the algorithm can also be implemented using a sampler for only the 0-coset and noise parameter bs_0 by bucketing the samples from different cosets in intermediate buffers. This approach has the advantage of being potentially simpler, but can make constant time implementations more troublesome (see Section 7.4).

Since we want to generate a large number of samples, our main criteria for the suitability of an algorithm is its expected running time. For any algorithm, this is lower bounded by the entropy of $\mathscr{D}_{\mathbb{Z},c,s}$, so a natural choice is (lazy) inversion sampling [63] or Knuth-Yao [21], since both are (close to) randomness optimal and their running time is essentially the number of random bits they consume, hence providing us with an optimal algorithm for our purpose. In fact, Knuth-Yao is a little faster than inversion sampling, so we focus on that.

Setup of Experimental Study

There are a number of cryptographic applications for our sampler, most of which use an integer sampler in one of three typical settings.

- The output distribution is the centered discrete Gaussian with fixed noise parameter.

This is the case in most basic LWE based schemes, where the noise for the LWE instance is sampled using an integer sampler.

- The output distribution is the discrete Gaussian with fixed noise parameter, but varying center. This is the case in the online phase of Peikert's sampler [63]. In particular, if applied to q-ary lattices the centers are restricted to the set $\frac{1}{q}\mathbb{Z}$.

- The output distribution is the discrete Gaussian where both, the center and the noise parameter may vary for each sample. This is typically used as a subroutine for sampling from the discrete Gaussian over lattices, as the GPV sampler [28] or in the offline phase of Peikert's sampler.

The ideas presented in this work can be applied to any of these settings. In particular, the algorithms in this chapter can be used to achieve new time-memory trade-offs in all three cases. The optimal trade-off is highly application specific and depends on a lot of factors, for example, the target platform (hardware vs. software), the cost of randomness (TRNGs vs. PRNGs), available memory, cost of evaluating $\exp(\cdot)$, cost of basic floating point/integer arithmetic, etc. In the following we present an experimental comparison of our algorithm to previous algorithms. Obviously, we are not able to take all factors into account, so we restrict ourselves to a comparison in a software implementation, where all algorithms use the same source of randomness (`NTL`'s PRNG), evaluate the randomness bit by bit in order to minimize randomness consumption, and use only elementary data types during the sampling. In particular, whenever FP arithmetic is necessary or $\rho_s(\cdot)$ needs to be evaluated during the sampling, all the algorithms use only double or extended double precision. This should be sufficient since we are targeting around 100 bits of security and the arguments in Chapter 6 apply to any algorithm. We do not claim that the implementation is optimal for any of the evaluated algorithms, but it should provide a fair comparison. We instantiated our algorithms with the parameters as

listed at the end of Sect. 7.3. Our implementation makes no effort towards a constant-time implementation. Even though turning Algorithm 7.1 into a constant-time algorithm is conceptually simple (cf. Sect. 7.4), this still requires a substantial amount of design and implementation effort, which is out of the scope of this work.

When referring to specific settings of the parameter s, we will often refer to it as multiple of $\sqrt{2\pi}$. The reason is that two slightly different definitions of $\rho_s(\cdot)$ are common in the literature and the factor $\sqrt{2\pi}$ converts between them. While we found one of them to be more convenient in the analytic part of this work, most previous experimental studies [14, 65] use the other. So this notation is for easier comparability.

Fixed Centered Gaussian

In this section we consider the simplest scenario for discrete Gaussian sampling: sampling from the centered discrete Gaussian distribution above a certain noise level. This is accomplished by Algorithm 7.2. Note that the parameter s_0 allows for a time memory trade-off in our setting: the larger s_0, the more memory required by our base sampler (Knuth-Yao), but the fewer the levels of recursion. More precisely, the memory requirement grows linearly with s_0, while the running time decreases logarithmically.

We compare the method in different settings to the only other adjustable time-memory trade-off known to date – the discrete Ziggurat. For our evaluation we modified the implementation of [14] to use elementary data types only during the sampling (as opposed to arbitrary precision arithmetic in the original implementation). The baseline algorithms in this setting are the Bernoulli-type sampler and Karney's algorithm, as they allow to sample from the centered discrete Gaussian quite efficiently using very little or no memory. Figure 7.1 shows the result of our experimental analysis for a set of representative s's. We chose the examples mostly according to the examples in [14], where we skipped the data point at $s = 10\sqrt{2\pi}$, since this is already a very narrow distri-

bution which can be efficiently sampled using Knuth-Yao with very moderate memory requirements. Instead, we show the results for $s = 2^{14}\sqrt{2\pi}$ (chosen somewhat arbitrarily), additionally to data points close to the ones presented in [14]: $s \in \{2^5, 2^{10}, 2^{17}\}\sqrt{2\pi}$.

Figure 7.1 shows that the two algorithms complement each other quite nicely: while Ziggurat allows for better trade-offs in the low memory regime, using convolution achieves much better running times in the high memory regime. This suggests that Ziggurat might be the better choice for constrained devices, but recall that it requires evaluations of $\exp(\cdot)$. So if s is not too large, even for constrained devices the convolution type sampler can be a better choice (see for example [65]).

Note that the improvement gained by using more memory deteriorates in our implementation, up to the point where using more memory actually hurts the running time (see Fig. 7.1, bottom right). A similar effect can be observed with the discrete Ziggurat algorithm. At first sight this might be counter-intuitive, but can be easily explained with a limited processor cache size: larger memory requirement in our case means fewer cache hits, which results in more RAM accesses, which are much slower. This nicely illustrates how dependent this trade-off is on the speed of the available memory. Since fast memory is usually much more expensive than slower memory, for a given budget it is very plausible that the money is better spent on limited amounts of fast memory and using Algorithm 7.2 rather than implementing the full Knuth-Yao with larger and slower memory. In our specific example (Fig. 7.1, bottom right), this means that using a convolution of two samples generated by smaller Knuth-Yao samplers is actually faster than generating the samples directly with a large Knuth-Yao sampler.

Fixed Gaussian with Varying Center

We now turn to the second setting, where the noise parameter is still fixed but the center may vary. In order to take advantage of the fact that the noise parameter is

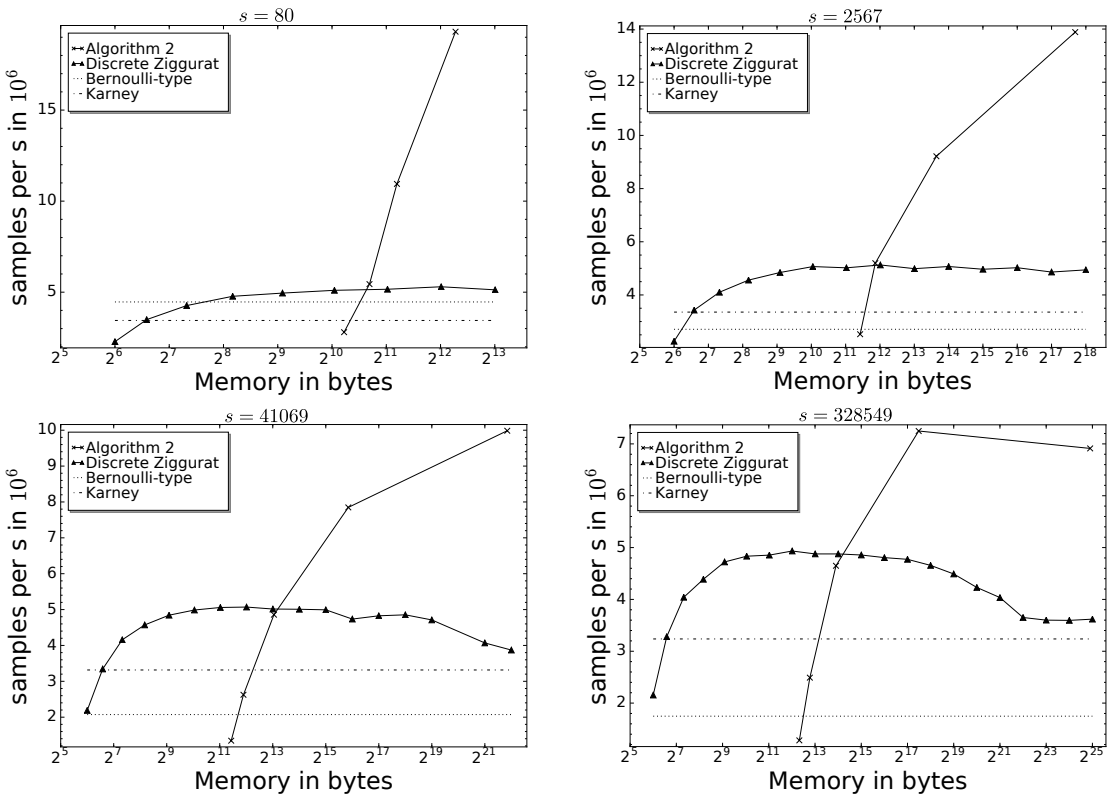


Figure 7.1. Time memory trade-off for Algorithm 7.2 and discrete Ziggurat compared to Bernoulli-type sampling and Karney's algorithm for $s \in \{2^5, 2^{10}, 2^{14}, 2^{17}\}\sqrt{2\pi}$. Knuth-Yao corresponds to right most point of Algorithm 7.2.

fixed and the center in a restricted set for the online phase, Peikert suggested that "if q is reasonably small it may be worthwhile (for faster rounding) to precompute the tables of the cumulative distribution functions for all q possibilities" [63]. This might be feasible, but only for very small q and s (depending on the available memory). If not enough memory is available, there is currently no option other than falling back to Karney's algorithm or rejection sampling.

Depending on the cost of randomness, speed and amount of available memory and processor speed for arithmetic, Knuth-Yao can be significantly faster than Karney's algorithm. For example, in our prototype implementation, Knuth-Yao was up to 6 times faster, but keep in mind that this number is highly platform dependent and can vary widely.

Accordingly, we can afford to invoke Knuth-Yao several times, sacrificing some running time for memory savings, and still outperform Karney's algorithm. Our algorithms offer exactly this kind of trade-off. There are two ways in which we can take advantage of convolution theorems to address the challenge of having to store q Knuth-Yao samplers. The first simply consists in storing the samplers for some smaller s_0, which will reduce the required memory by a factor s/s_0. After obtaining a sample from the right coset, using only the 0-coset we can generate and add a sample from a wider distribution to obtain the correct distribution. This is very similar to Algorithm 7.2 with the additional step of adding a sample from the right coset, where we simply invoke Corollary 7 once more. This step will increase the running time by at most $\log_{s_0} s$ additively (cf. Lemma 22).

Note that there is a limit to this technique though, since we need $s_0 > \sqrt{2}\eta_\varepsilon(\mathbb{Z})$ for the convolution to yield the correct output distribution. If s is already small, but there is not enough memory available because q is too large, this approach will fail. In this case we can use the algorithm from Sect. 7.2 to reduce the number of samplers needed to be stored. In particular, for any base b such that[3] $\mathrm{rad}(q) \mid b$, we can cut down on the memory cost by a factor q/b, which will increase the running time by $\lceil \log_b q \rceil$. For this, we simply need to express the center c in the base b and round the digits individually using $\textsc{SampleC}_b$. For example, if q is a power of a small prime p, we can choose b to be any multiple of p. This can dramatically increase the modulus q for which we can sample fast with a given amount of memory, assuming $\mathrm{rad}(q)$ is small. As a more specific example, say q is a perfect square and let $b = \sqrt{q}$. Instead of storing q Knuth-Yao samplers and invoking one when a sample is required for a coset $\frac{1}{q}\mathbb{Z}$, we can store b samplers and randomly round each of the 2 digits of the center in base b successively. This effectively doubles the running time, but this is likely to still be much faster than

[3]This is the condition for $\frac{1}{q}$ being expressible as a finite number in base b.

Karney's algorithm (again, depending on the platform), but we reduced the amount of necessary memory by a factor $\sqrt{q}$.

Clearly, depending on the specific q, s and platform, the two techniques can be combined. The optimal trade-off depends on all three factors and has to be evaluated for each application. Our algorithms provide developers with the tools to optimize this trade-off and make the most of the available resources.

Varying Gaussian

Finally, we evaluate the practical performance of our full sampler, SAMPLEZ. Precomputing the value K, as suggested in Sect. 7.3, made little difference in our software implementation and we show results for the algorithm that does not precompute K. The bottleneck in our algorithm is the call to SAMPLEC, as it consumes a number of samples which depends on the base b. Again, similar to the previous section, the base b offers a time-memory trade-off, which is the target of our evaluation. We experimented with the sampler for a wide range of noise parameters s, but since our algorithm is essentially independent of s (as long as it is $\leq s_{\max}$), it is not surprising that the trade-off is essentially the same in all cases. Accordingly, we present only one exemplary result in Fig. 7.2. As a frame of reference, rejection sampling achieved $0.994 \cdot 10^6$ samples per second, which shows that by spending only very moderate amounts of memory ($<$ 1mb), our algorithm can match and outperform rejection sampling. On the other hand, Karney's algorithm achieved $3.281 \cdot 10^6$ samples per second, which seems out of reach for reasonable amounts of memory, making it the most efficient choice in this setting, if no other criteria are of concern. But we stress again that this depends highly on how efficiently Knuth-Yao can be implemented compared to Karney's algorithm on the target platform. While the running time of both, rejection sampling and Karney's algorithm depends on s, this dependence is rather weak (logarithmic with small constants) so the picture does not

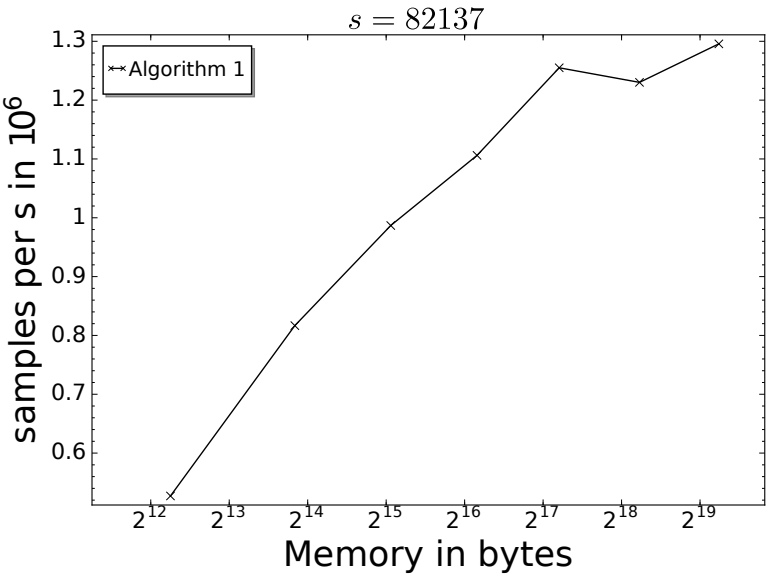


Figure 7.2. Time memory trade-off Algorithm 7.1 for $s = 2^{15}\sqrt{2\pi}$.

change much for other noise parameters.

Recall that our algorithm can be split into online and offline phase, since the base samples are independent of the target distribution. Karney's algorithm also initially samples from a Gaussian that is independent of the target distribution, so a similar approach can be applied. However, the trade-off is fixed and no speed-ups can be achieved by spending more memory.

We tested both algorithms, where we assumed that the offline phase is free, for a wide range of s. For this, we fixed $b = 16$ for our algorithm, which seemed to be a good choice in our setting. Note that similar to Sect. 7.5.4, spending more memory (and increasing b) should in theory only improve the algorithm. But if this comes at the cost of slowing down memory access due to a limited cache size, this can actually hurt performance. The results are depicted in Fig. 7.3. The graph allows for two interesting observations: First, our algorithm consistently outperforms Karney's algorithm in this setting. So if the offline phase can be considered to be free or a limited life-time is acceptable (cf. Sect. 7.4), our algorithm seems to be the better choice. Second, as

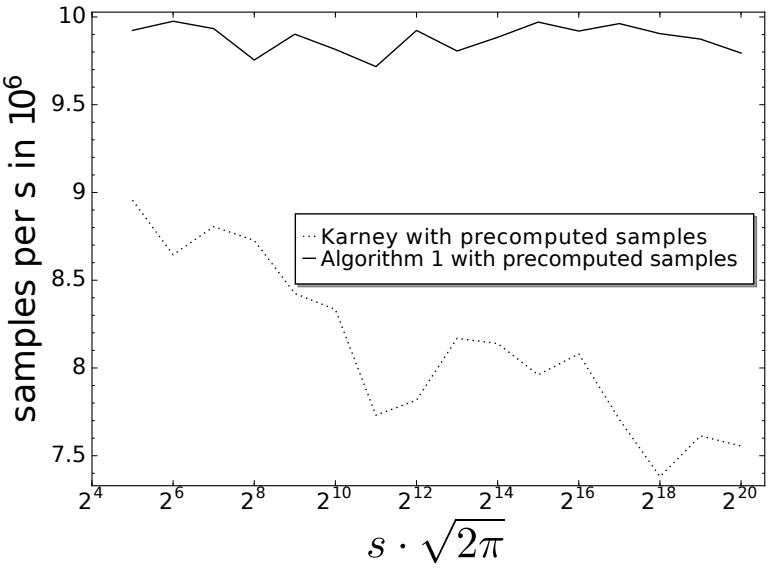


Figure 7.3. Performance of Algorithm 7.1 compared to Karney's algorithm, (online phase only).

expected, our algorithm is essentially independent of s (as long as it is $< s_{\max}$), while the performance of Karney's algorithm deteriorates as s grows. This is due to the fact that Karney's algorithm requires to sample a uniform number in $[0, s]$ during the online phase, which is logarithmic in s. This leads to a larger gap between the performance of the two algorithms as s grows, and supports the claim that our sampler allows for an efficient constant time implementation. In contrast, both Karney's algorithm and rejection sampling seem to be inherently costly to turn into constant time algorithms, due to their dependence on s and the fact that they are probabilistically rejecting samples.

In summary, we believe that there are a number of applications and target platforms, where our algorithm will be the best choice to implement a discrete Gaussian sampler.

Chapter 7, in full, is a reprint of material as it appears (with minor modifications) in "Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time" by Daniele Micciancio and Michael Walter, published in the proceedings of the Thirty-Seventh

Annual International Cryptology Conference (CRYPTO 2017). The dissertation author was the primary investigator and author of this paper.

Bibliography

[1] M. R. Albrecht, C. Cocis, F. Laguillaumie, and A. Langlois. Implementing candidate graded encoding schemes from ideal lattices. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 752–775. Springer, Heidelberg, Nov. / Dec. 2015.

[2] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3):169–203, Oct. 2015.

[3] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange - A new hope. In *USENIX Security Symposium*, pages 327–343. USENIX Association, 2016.

[4] L. Bahler, G. D. Crescenzo, Y. Polyakov, K. Rohloff, and D. B. Cousins. Practical implementation of lattice-based program obfuscators for point functions. In *HPCS 2017: International Conference on High Performance Computing & Simulation*, pages 761–768. IEEE, 2017.

[5] S. Bai, A. Langlois, T. Lepoint, D. Stehlé, and R. Steinfeld. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 3–24. Springer, Heidelberg, Nov. / Dec. 2015.

[6] M. Bellare, M. Fischlin, S. Goldwasser, and S. Micali. Identification protocols secure against reset attacks. In B. Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 495–511. Springer, Heidelberg, May 2001.

[7] D. J. Bernstein and T. Lange. Non-uniform cracks in the concrete: The power of free precomputation. In K. Sako and P. Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 321–340. Springer, Heidelberg, Dec. 2013.

[8] J. Blömer. Closest vectors, successive minima, and dual HKZ-bases of lattices. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *ICALP 2000: 27th International*

Colloquium on Automata, Languages and Programming, volume 1853 of *Lecture Notes in Computer Science*, pages 248–259. Springer, Heidelberg, July 2000.

[9] X. Boyen. Attribute-based functional encryption on lattices. In A. Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 122–142. Springer, Heidelberg, Mar. 2013.

[10] X. Boyen and Q. Li. Attribute-based encryption for finite automata from LWE. In M. H. Au and A. Miyaji, editors, *ProvSec 2015: 9th International Conference on Provable Security*, volume 9451 of *Lecture Notes in Computer Science*, pages 247–267. Springer, Heidelberg, Nov. 2015.

[11] Z. Brakerski and V. Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, Heidelberg, Aug. 2011.

[12] Z. Brakerski, V. Vaikuntanathan, H. Wee, and D. Wichs. Obfuscating conjunctions under entropic ring LWE. In M. Sudan, editor, *ITCS 2016: 7th Innovations in Theoretical Computer Science*, pages 147–156. Association for Computing Machinery, Jan. 2016.

[13] L. G. Bruinderink, A. Hülsing, T. Lange, and Y. Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In B. Gierlichs and A. Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, volume 9813 of *Lecture Notes in Computer Science*, pages 323–345. Springer, Heidelberg, Aug. 2016.

[14] J. Buchmann, D. Cabarcas, F. Göpfert, A. Hülsing, and P. Weiden. Discrete ziggurat: A time-memory trade-off for sampling from a Gaussian distribution over the integers. In T. Lange, K. Lauter, and P. Lisonek, editors, *SAC 2013: 20th Annual International Workshop on Selected Areas in Cryptography*, volume 8282 of *Lecture Notes in Computer Science*, pages 402–417. Springer, Heidelberg, Aug. 2014.

[15] Y. Chen. *Réduction de réseau et sécurité concréte du chiffrement complétement homomorphe*. PhD thesis, ENS Paris, 2013.

[16] Y. Chen and P. Q. Nguyen. BKZ 2.0: Better lattice security estimates. In D. H. Lee and X. Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, Heidelberg, Dec. 2011.

[17] W. Dai, Y. Doröz, Y. Polyakov, K. Rohloff, H. Sajjadpour, E. Savaş, and B. Sunar. Implementation and evaluation of a lattice-based key-policy ABE scheme. Cryptology ePrint Archive, Report 2017/601, 2017. http://eprint.iacr.org/2017/601.

[18] Y. Dodis and J. P. Steinberger. Message authentication codes from unpredictable block ciphers. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 267–285. Springer, Heidelberg, Aug. 2009.

[19] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal Gaussians. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 40–56. Springer, Heidelberg, Aug. 2013.

[20] L. Ducas and D. Micciancio. FHEW: Bootstrapping homomorphic encryption in less than a second. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, Heidelberg, Apr. 2015.

[21] N. C. Dwarakanath and S. D. Galbraith. Sampling from discrete gaussians for lattice-based cryptography on a constrained device. *Appl. Algebra Eng. Commun. Comput.*, 25(3):159–180, 2014.

[22] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44(170):463–471, Apr. 1985.

[23] N. Gama and P. Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In R. E. Ladner and C. Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 207–216. ACM Press, May 2008.

[24] N. Gama and P. Q. Nguyen. Predicting lattice reduction. In N. P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, Heidelberg, Apr. 2008.

[25] N. Gama, P. Q. Nguyen, and O. Regev. Lattice enumeration using extreme pruning. In H. Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 257–278. Springer, Heidelberg, May 2010.

[26] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17. Springer, Heidelberg, May 2013.

[27] C. Gentry and S. Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In K. G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, Heidelberg, May 2011.

[28] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In R. E. Ladner and C. Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, May 2008.

[29] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32. ACM Press, May 1989.

[30] D. Goldstein and A. Mayer. On the equidistribution of Hecke points. *Forum Mathematicum*, 15(2):165–189, Jan. 2003.

[31] K. D. Gür, Y. Polyakov, K. Rohloff, G. W. Ryan, and E. Savaş. Implementation and evaluation of improved gaussian sampling for lattice trapdoors. Cryptology ePrint Archive, Report 2017/285, 2017. http://eprint.iacr.org/2017/285.

[32] G. Hanrot, X. Pujol, and D. Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In P. Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 447–464. Springer, Heidelberg, Aug. 2011.

[33] G. Hanrot and D. Stehlé. Improved analysis of kannan's shortest lattice vector algorithm. In A. Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 170–186. Springer, Heidelberg, Aug. 2007.

[34] G. Hanrot and D. Stehlé. Improved analysis of kannan's shortest lattice vector algorithm. *CoRR*, abs/0705.0965, 2007.

[35] B. Helfrich. Algorithms to construct Minkowski reduced and Hermite reduced lattice bases. *Theoretical Computer Science*, 41:125 – 139, 1985.

[36] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *15th Annual ACM Symposium on Theory of Computing*, pages 193–206. ACM Press, Apr. 1983.

[37] C. F. F. Karney. Sampling exactly from the normal distribution. *ACM Trans. Math. Softw.*, 42(1):3:1–3:14, 2016.

[38] S. Khot. Hardness of approximating the shortest vector problem in lattices. In *45th Annual Symposium on Foundations of Computer Science*, pages 126–135. IEEE Computer Society Press, Oct. 2004.

[39] T. Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In R. Gennaro and M. J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 3–22. Springer, Heidelberg, Aug. 2015.

[40] A. Langlois, D. Stehlé, and R. Steinfeld. GGHLite: More efficient multilinear maps from ideal lattices. In P. Q. Nguyen and E. Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 239–256. Springer, Heidelberg, May 2014.

[41] H. j. Lenstra, A. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[42] L. A. Levin. Randomness and non-determinism. *Journal of Symbolic Logic*, 58:1102–1103, 1993.

[43] J. Li and W. Wei. Slide reduction, successive minima and several applications. *Bulletin of the Australian Mathematical Society*, 88:390–406, 2013.

[44] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In A. Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, Heidelberg, Feb. 2011.

[45] M. Liu and P. Q. Nguyen. Solving BDD by enumeration: An update. In E. Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 293–309. Springer, Heidelberg, Feb. / Mar. 2013.

[46] Z. Liu, H. Seo, S. S. Roy, J. Großschädl, H. Kim, and I. Verbauwhede. Efficient ring-LWE encryption on 8-bit AVR processors. In T. Güneysu and H. Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, volume 9293 of *Lecture Notes in Computer Science*, pages 663–682. Springer, Heidelberg, Sept. 2015.

[47] C. M. Mayer. Implementing a toolkit for ring-LWE based cryptography in arbitrary cyclotomic number fields. Cryptology ePrint Archive, Report 2016/049, 2016. http://eprint.iacr.org/2016/049.

[48] D. Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. In *39th Annual Symposium on Foundations of Computer Science*, pages 92–98. IEEE Computer Society Press, Nov. 1998.

[49] D. Micciancio. Inapproximability of the shortest vector problem: Toward a deterministic reduction. *Theory of Computing*, 8(22):487–512, Sept. 2012.

[50] D. Micciancio and C. Peikert. Hardness of SIS and LWE with small parameters. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 21–39. Springer, Heidelberg, Aug. 2013.

[51] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. In *45th Annual Symposium on Foundations of Computer Science*, pages 372–381. IEEE Computer Society Press, Oct. 2004.

[52] D. Micciancio and M. Walter. Fast lattice point enumeration with minimal overhead. In P. Indyk, editor, *26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 276–294. ACM-SIAM, Jan. 2015.

[53] D. Micciancio and M. Walter. Practical, predictable lattice basis reduction. Cryptology ePrint Archive, Report 2015/1123, 2015. http://eprint.iacr.org/2015/1123.

[54] D. Micciancio and M. Walter. Practical, predictable lattice basis reduction. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 820–849. Springer, Heidelberg, May 2016.

[55] D. Micciancio and M. Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. In J. Katz and H. Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 455–485. Springer, Heidelberg, Aug. 2017.

[56] D. Micciancio and M. Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. Cryptology ePrint Archive, Report 2017/259, 2017. http://eprint.iacr.org/2017/259.

[57] I. Mironov. Renyi differential privacy. *CoRR*, abs/1702.07476, 2017.

[58] A. Neumaier. Bounding basis reduction properties. *Designs, Codes and Cryptography*, 84(1):237–259, July 2017.

[59] P. Q. Nguyen. Hermite's constant and lattice algorithms. Information Security and Cryptography, pages 19–69. Springer, Heidelberg, 2010.

[60] P. Q. Nguyen and O. Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In S. Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288. Springer, Heidelberg, May / June 2006.

[61] P. Q. Nguyen and D. Stehlé. LLL on the average. In F. Hess, S. Pauli, and M. Pohst, editors, *ANTS 2006: 7th International Algorithmic Number Theory Symposium*, Lecture Notes in Computer Science, pages 238–256. Springer, Heidelberg, July 2006.

[62] P. Q. Nguyen and J. Stern. Lattice reduction in cryptology: An update. In W. Bosma, editor, *ANTS 2000: 4th International Algorithmic Number Theory Symposium*, volume 1838 of *Lecture Notes in Computer Science*, pages 85–112. Springer, Heidelberg, July 2000.

[63] C. Peikert. An efficient and parallel Gaussian sampler for lattices. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97. Springer, Heidelberg, Aug. 2010.

[64] P. Pessl. Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In O. Dunkelman and S. K. Sanadhya, editors, *Progress in Cryptology - INDOCRYPT 2016: 17th International Conference in Cryptology in India*, volume 10095 of *Lecture Notes in Computer Science*, pages 153–170. Springer, Heidelberg, Dec. 2016.

[65] T. Pöppelmann, L. Ducas, and T. Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 353–370. Springer, Heidelberg, Sept. 2014.

[66] T. Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. Cryptology ePrint Archive, Report 2017/480, 2017. http://eprint.iacr.org/2017/480.

[67] K. Rohloff and D. B. Cousins. A scalable implementation of fully homomorphic encryption built on NTRU. In R. Böhme, M. Brenner, T. Moore, and M. Smith, editors, *FC 2014 Workshops*, volume 8438 of *Lecture Notes in Computer Science*, pages 221–234. Springer, Heidelberg, Mar. 2014.

[68] S. S. Roy, O. Reparaz, F. Vercauteren, and I. Verbauwhede. Compact and side channel secure discrete Gaussian sampling. Cryptology ePrint Archive, Report 2014/591, 2014. http://eprint.iacr.org/2014/591.

[69] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede. Compact ring-LWE cryptoprocessor. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 371–391. Springer, Heidelberg, Sept. 2014.

[70] M.-J. O. Saarinen. Gaussian sampling precision in lattice cryptography. Cryptology ePrint Archive, Report 2015/953, 2015. http://eprint.iacr.org/2015/953.

[71] M.-J. O. Saarinen. Arithmetic coding and blinding countermeasures for lattice signatures. *Journal of Cryptographic Engineering*, pages 1–14, 2017.

[72] C. P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2-3):201–224, Aug. 1987.

[73] C.-P. Schnorr. Lattice reduction by random sampling and birthday methods. In H. Alt and M. Habib, editors, *STACS 2003: 20th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2607 of *Lecture Notes in Computer Science*, pages 145–156. Springer, Heidelberg, Feb. 2003.

[74] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66(1):181–199, Aug. 1994.

[75] V. Shoup. NTL: A library for doing number theory. Available at http://www.shoup.net/ntl/, 2017.

[76] K. Takashima and A. Takayasu. Tighter security for efficient lattice cryptography via the Rényi divergence of optimized orders. In M. H. Au and A. Miyaji, editors, *ProvSec 2015: 9th International Conference on Provable Security*, volume 9451 of *Lecture Notes in Computer Science*, pages 412–431. Springer, Heidelberg, Nov. 2015.

[77] The FPLLL development team. fplll, a lattice reduction library. Available at https://github.com/fplll/fplll, 2017.

[78] M. Walter. Bkz simulator. Available at http://cseweb.ucsd.edu/~miwalter/src/sim_bkz.sage, 2014.

[79] M. Walter. Lattice point enumeration on block reduced bases. In A. Lehmann and S. Wolf, editors, *ICITS 15: 8th International Conference on Information Theoretic Security*, volume 9063 of *Lecture Notes in Computer Science*, pages 269–282. Springer, Heidelberg, May 2015.

[80] L. Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.