IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 7, 2017, accepted August 22, 2017, date of publication September 1, 2017,
date of current version November 28, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2748179

A Practical View of the State-of-the-Art of Lattice-Based Cryptanalysis

ARTUR MARIANO[1], THIJS LAARHOVEN[2], FÁBIO CORREIA[3], MANUEL RODRIGUES[1], AND GABRIEL FALCÃO[1], (Senior Member, IEEE)

[1]Instituto de Telecomunicações and Department of Electrical and Computer Engineering, University of Coimbra, Pólo II, 3030-290 Coimbra, Portugal
[2]IBM Research Zurich, CH-8803 Ruschlikon, Switzerland
[3]eXXcellent Solutions, 89075 Ulm, Germany

Corresponding author: Artur Mariano (artur.mariano@co.it.pt)

This work was supported by the Instituto de Telecomunicações and Fundação para Ciência e a Tecnologia under Grant UID/EEA/50008/2013. The work of T. Laarhoven was supported by the SNSF ERC Transfer under Grant CRETP2-166734 FELICITY.

ABSTRACT This paper describes the lattice problems that are key in the study of lattice-based cryptography, identifies and categorizes methods for solving these problems, analyzes existing implementations of these algorithms, and extrapolates on the future of lattice-based cryptanalysis, based on the foreseeable advances in computer architecture. Some future lines of work are given, considering the existence of parallel architectures that seem adequate for current attacks.

INDEX TERMS Lattices, cryptanalysis, parallel.

I. INTRODUCTION

Cryptology is the science that studies both cryptography and cryptanalysis. ''Cryptography'' can be defined as the process of creating and understanding codes or systems that keep information secret, commonly for use in digital communication. Oftentimes, one refers to these codes and systems as ''schemes'' and ''cryptosystems'', terms that we also use throughout this paper, interchangeably. ''Cryptanalysis'' can be defined as the science that studies processes and methods to recover secrets when the key is unknown. While cryptography is the process of creating new systems, cryptanalysis, on the one hand, enables the analysis of existing cryptosystems, so that we either trust them or find out that these schemes are vulnerable to attacks. At the same time, cryptanalysis is an essential tool to define parameters for new cryptosystems, so that they are simultaneously secure and efficient.

In the mid-nineties, due to a breakthrough work of Shor on quantum algorithms, it turned out that most cryptosystems currently deployed in practice, such as RSA and ElGamal, are insecure against adversaries with access to quantum computers [21], [103], [104]. More specifically, the problems that underpin the security of these classical cryptosystems, such as factoring large numbers or computing discrete logarithms, become practical in the presence of large-scale quantum computers. Soon after, the community started to work towards ''post-quantum'' cryptosystems based on other computationally hard problems, which are presumably more resistant against attacks operated with quantum computers.

In the late 90s, Ajtai discovered that certain lattice problems have interesting properties for cryptography, such as worst-case/average-case reductions,[1] and that lattices can be used for building cryptosystems [6]. Since to date no fast quantum algorithms have been found to solve hard lattice problems efficiently, cryptography based on lattices is also a candidate for post-quantum cryptography. Today, lattice-based cryptosystems are becoming increasingly popular due to a variety of reasons, including supporting advanced cryptographic primitives such as fully homomorphic encryption,[2] its relative efficiency, and its expectancy to be safe against quantum adversaries [21], [72].

Lattices are discrete subgroups of the n-dimensional Euclidean space $\mathbb{R}^n$, with a strong periodicity property.[3] A lattice $\mathcal{L}$ generated by a basis $\mathbf{B}$, a set of linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$ in $\mathbb{R}^n$, is denoted by:

$$\mathcal{L}(\mathbf{B}) = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \sum_{i=1}^{m} \mathbf{u}_i \mathbf{b}_i,\ \mathbf{u} \in \mathbb{Z}^m \right\}, \qquad (1)$$

[1]Roughly speaking, this means that breaking cryptosystems based on randomly chosen, average-case lattice problem instances, is at least as hard as solving certain lattice problems in the worst case.

[2]A cryptosystem that supports Fully Homomorphic Encryption can implement any operation on encrypted data, without decrypting data. This is particularly useful when e.g. outsourcing sensitive computations on private data to a cloud server. The reader is referred to the survey [88] for a practical perspective of Fully Homomorphic Encryption.

[3]We refer the reader to the papers [82], [94] in order to learn more about lattices, especially in the context of lattice-based cryptography.

where $m \leq n$ is the *rank* of the lattice. When $m = n$, the lattice is said to be of *full rank*. When n is at least 2, each lattice has infinitely many different bases.

Although there are non-integer lattices, lattice-based cryptography commonly uses integer lattices: solving lattice problems on integer lattices is still hard, and integer lattices are easier to handle computationally (e.g. there are no precision/numerical problems). In the literature, there are various types of lattices, such as Goldstein-Mayer lattices (also referred to as *random lattices* [107]) and Ajtai lattices [5] (which typically have vectors with relatively small coordinates). Some lattices, such as ideal lattices, have additional structure [60]. Ideal lattices are particularly important in lattice-based cryptography because some cryptosystems take advantage of their additional structure to obtain smaller key sizes, which translates into improved efficiency for the resulting cryptosystems [61], [106]. Adversaries may however also be able to take advantage of this additional structure leading to weaker security, as discussed later in this paper.

As an example, Figure 1 shows a lattice in $\mathbb{R}^2$, where the basis $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2\}$. The vector $\mathbf{b}_3$ shown in the picture is a linear combination of the basis vectors. This linear combination also shows that $\mathbf{b}_1$ can be made shorter (in terms of Euclidean norm) at the cost of $\mathbf{b}_2$, given that $\mathbf{b}_3$ is smaller than $\mathbf{b}_1$. This process of making lattice vectors (bases) shorter by adding/subtracting other lattice vectors is often referred to as vector (basis) reduction, and is widely used in various lattice algorithms.

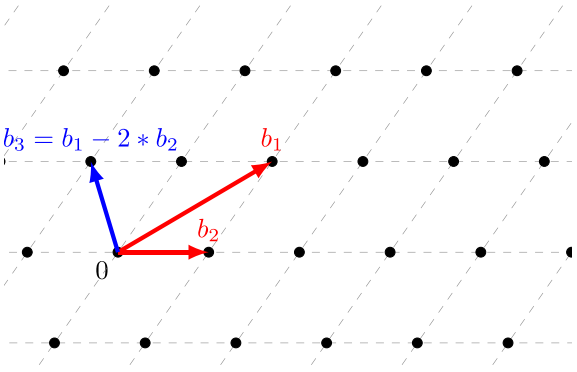

FIGURE 1. Example of a lattice in $\mathbf{R^2}$ and its basis ($\mathbf{b_1}$,$\mathbf{b_2}$) in red.

As described above, cryptosystems base their security on hard mathematical problems. For instance, the security of the RSA cryptosystem is based on the hardness of factoring certain large integers. Put roughly, if an attacker is able to efficiently factor such large numbers, with the same number of bits as the key, then he can break the system. The foundation of the security of RSA (and other cryptosystems) is therefore that, as factoring large integers is a computationally hard problem and its complexity grows fast with the input size, no attacker can factor a sufficiently large number in a reasonable amount of time and thus break the system.

The security of lattice-based cryptosystems is also based on hard mathematical problems, but problems that are built on lattices. These include 1) lattice basis reduction, 2) the Shortest Vector Problem (SVP), 3) the Closest Vector Problem (CVP), 4) the Learning with Errors (LWE) and several variants of these. There are many other problems that lattice-based cryptosystems base their security on, but these are perhaps the most relevant to cover in a practical survey. Here the input size commonly scales with the lattice dimension: in low dimensions (say $n \leq 100$), solving these problems remains feasible in practice, but when the lattice dimension becomes very large (e.g. $n \geq 10000$), these problems become intractable with current methods.

Due to the connection between the problems and the security of the corresponding cryptosystems, the algorithms that solve these problems are sometimes referred to as *attacks*. The main reason why it is important to study the potential of attacks is because they provide insights for parameter selection (e.g. key sizes) of lattice-based cryptographic primitives. Parameter selection is typically a two-phase process: first, the community assesses the practical potential of attacks, implemented on high-end computer architectures. Then, the parameters for the cryptosystems are selected such that reaching such parameters becomes impractical, based on the results of the first phase. This has to be the case even when the best attacks are implemented on the highest-end computer architectures. This is why the last years have witnessed considerable efforts towards efficient implementation of attacks. Implementing the best attacks to date on modern, parallel multi- and many-core systems provides a solid idea of how hard it is to solve lattice problems, and how to accurately select parameters for lattice-based cryptosystems guaranteeing both security and efficiency.

A. CONTRIBUTIONS

In this survey, we identify how these high-end parallel processors can be used to efficiently solve the problems that underpin the security of lattice-based cryptosystems, and what are the main challenges found when trying to attain full performance in such scenarios.

To this end, we gather, categorize and analyze the advances on the practical side of lattice-based cryptanalysis, with focus on lattice basis reduction, the SVP, the CVP and the LWE problem. After gathering the most relevant implementations, and categorizing them in various classes of solvers, we report on their throughput performance and scalability, based on results from the literature. At the end we further provide references to various tools and libraries available online that can be used to further assess the practical performance of various attacks.

B. ROADMAP

The remainder of this paper is organized as follows. Section II introduces the lattice problems that lay at the the foundation of lattice-based cryptography. In Section III, we identify and categorize the algorithms for solving the problems presented in Section II. Section IV gathers the work around the implementations of these attacks, and presents and comments

on their throughput performance and scalability on parallel architectures. Section V describes libraries and software modules that contain implementations of the attacks mentioned in Section III and other algorithms that are useful in the context of lattice-based cryptanalysis. Section VI discusses so-called ''challenges'' which are available online, that can be used for evaluating the performance of algorithms, as well as for assessing the security of lattice-based cryptosystems. Finally, Section VII extrapolates on the future of lattice-based cryptanalysis, based on the foreseeable advances in computer architecture, and provides some future lines of work.

II. LATTICE PROBLEMS USED IN CRYPTOSYSTEMS

This section introduces some of the most relevant problems underlying the security of lattice-based cryptography. Although there is a large variety of hard lattice problems, this survey will focus on lattice basis reduction, the SVP, the CVP, and the LWE problem, as these are among the most important in the context of lattice-based cryptanalysis. For a broader compilation of lattice problems, including the Shortest Integer Solution (SIS) problem, the reader is referred to the surveys [70], [72], [93]. The rest of this section introduces the aforementioned four lattice problems and some of their relaxed versions, i.e. problems where the solutions are allowed to be slightly off, by a given amount, from the exact solutions to these problems.

A. LATTICE BASIS REDUCTION

Although lattice basis reduction is typically not mentioned when enumerating hard lattice problems, it is a prime process in this context. Lattice basis reduction is the process of transforming a given lattice basis $\mathbf{B}$ into another lattice basis $\mathbf{B}'$ of the same lattice (i.e., $\mathcal{L}(\mathbf{B}) = \mathcal{L}(\mathbf{B}')$), whose vectors are shorter and more orthogonal than those of $\mathbf{B}$. Figure 2 shows a basis transformation in order to reduce the original basis $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2)$ into a shorter, more orthogonal basis $\mathbf{B}' = (\mathbf{b}_3, \mathbf{b}_4)$.

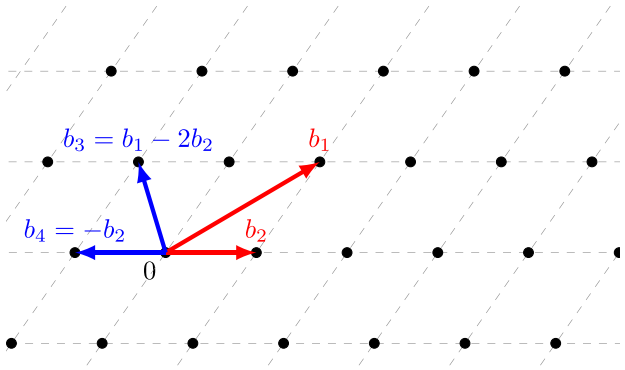

FIGURE 2. Example of a lattice basis reduction process in $\mathbb{R}^2$. The original basis $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2)$ is drawn in red and the resultant basis $\mathbf{B}' = (\mathbf{b}_3, \mathbf{b}_4)$ is in blue.

All too often, reduced bases are called ''superior'' bases. There is not a universal definition for the quality of lattice bases, but Mariano has recently enumerated in his PhD thesis a series of metrics that can be used to assess the quality of lattice bases (see [63], Chapter 2, Section 2.1.1.). Intuitively, all these metrics aim to capture the main goals of lattice basis reduction: to make the basis vectors as short and orthogonal as possible.

Lattice basis reduction is a well studied problem, and there are several practical algorithms for reducing lattice bases, such as the Lenstra-Lenstra-Lovász (LLL) algorithm [57] and the Block Korkine-Zolotarev (BKZ) algorithm [98]. Lattice basis reduction is relevant in the context of lattice-based cryptanalysis as there is a strong correlation between the quality of the basis and the performance of SVP- and CVP-solvers, which we present in the following. Moreover, approximate versions of these problems (in particular the SVP) can be solved with lattice basis reduction algorithms.

B. THE SHORTEST VECTOR PROBLEM (SVP)

Formally, the SVP can be defined as: given a basis $\mathbf{B}$ of the lattice $\mathcal{L}$, find a non-zero vector $\mathbf{p} \in \mathcal{L}$ such that: $\|\mathbf{p}\| = \min\{\|\mathbf{v}\| : \mathbf{v} \in \mathcal{L}(\mathbf{B}), \|\mathbf{v}\| \neq 0\}$. The norm of such a shortest non-zero vector[4] of a lattice is sometimes denoted by $\lambda_1(\mathcal{L})$. In 1998, Ajtai showed that for the Euclidean norm, solving the SVP is NP-hard under randomized reductions [6]. Figure 3 shows the shortest vectors in the lattice of Figures 1 and 2, which are the solutions for the SVP.

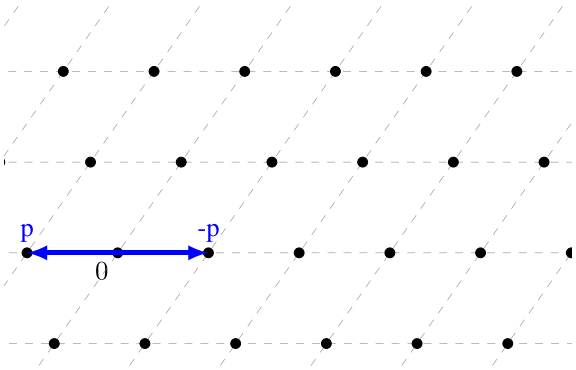

FIGURE 3. Example of the SVP, on a lattice in $\mathbb{R}^2$. Shortest vectors in blue.

The above is the definition of the *exact* version of the SVP; in an approximate version of the problem, typically referred to as α-SVP, the solution is a vector whose norm is, at most, a factor $\alpha \geq 1$ bigger than $\lambda_1(\mathcal{L})$. For a comprehensive review of the variants and approximate versions of the SVP, the reader is referred to [72]. As an example, the Ajtai-Dwork cryptosystem, one of the earliest cryptosystems based on lattices, bases its security on the γ-Unique SVP, a problem that derives from the SVP [7].

The cryptographic community has developed in-depth knowledge of the SVP. However, this knowledge is primarily theoretical, and the practical performance of these algorithms

[4]Note that due to the natural symmetry in lattices, there is not only one shortest vector.

has only started to be seriously studied over the last years. In fact, there remain many gaps in knowledge, particularly regarding the difference of performance between the theoretical expectations and actual practical performance of many SVP-solvers. This is problematic, as the security parameters of cryptosystems are selected based upon the practical performance of attacks, which theoretical expectations are often not congruent with.

C. THE CLOSEST VECTOR PROBLEM (CVP)

Formally, the CVP can be defined as: given a basis $\mathbf{B}$ of a lattice $\mathcal{L}$, and a target vector $\mathbf{t}$, find a lattice vector $\mathbf{p}$ that is closest to $\mathbf{t}$, i.e. such that: $\|\mathbf{p}-\mathbf{t}\| = \min\{\|\mathbf{w}-\mathbf{t}\| : \mathbf{w} \in \mathcal{L}(\mathbf{B})\}$. This problem was shown to be NP-hard by Van Emde Boas [26]. Figure 4 shows an example of the CVP, where the target vector is in red and the solution is in blue.

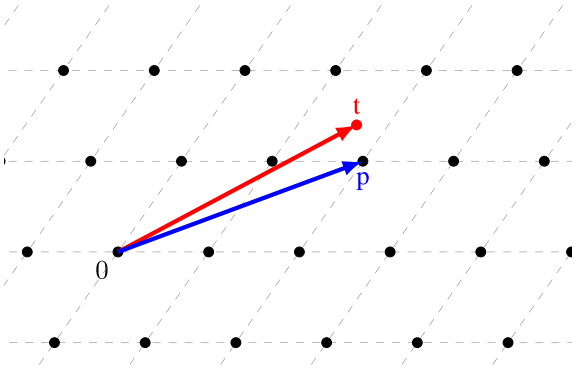

FIGURE 4. Example of the CVP, on a lattice in R^2. Target vector t in red, solution in blue.

Like the SVP, the CVP also has derivative problems and approximate versions (see [70], [72] for additional details). Cryptosystems whose security is based on the hardness of the CVP, directly or indirectly, include, among others, the Goldreich-Goldwasser-Halevi (GGH) cryptosystem [46].

The CVP is also a very well studied problem (even though the SVP has been attracting more attention over the past years) and there are various algorithms to solve exact and approximate versions of the CVP, some of which are very closely related to SVP solvers.

D. THE LEARNING WITH ERRORS (LWE) PROBLEM

Having been introduced in 2005 by Regev [95], the LWE problem is more recent than the previous problems and is a generalization of the Learning Parity with Noise (LPN) problem [90]. The parameters of this problem are the (lattice) dimension $n \geq 1$, an integer modulus $q \geq 2$, and an error distribution χ over $\mathbb{Z}_q$. Intuitively, χ can be thought of as a narrow (discretized) Gaussian centered around 0. In essence, there are two main versions of this problem [93], [95]: a search version and a decision version. In both these problems, one is given samples of the form $(\mathbf{a}_i, b_i)$ where the vectors $\mathbf{a}_i \in \mathbb{Z}_q^n$ are chosen uniformly at random, and the scalars b_i are of the form $b_i = \langle \mathbf{a}_i, \mathbf{s}\rangle + e_i \mod q$ where $\mathbf{s} \in \mathbb{Z}_q^n$ is a (fixed) secret vector and the $e_i \in \mathbb{Z}_q$ are small hidden "errors" chosen according to the distribution χ. In the search version, the problem is to find $\mathbf{s}$, given (arbitrarily) many samples $(\mathbf{a}_i, b_i)$ (without knowledge of the e_i's). In the decision version, the problem is to decide whether a set of provided samples $(\mathbf{a}_i, b_i)$ is indeed of the above form, or that the b_i were in fact chosen uniformly at random from $\mathbb{Z}_q$.

For the LWE problem, the link with lattices is not as immediate as for the SVP and the CVP. Given several samples $(\mathbf{a}_i, b_i)$, we can gather them into $(\mathbf{A}, \mathbf{b})$ where $\mathbf{A}$ is a matrix with the vectors $\mathbf{a}_i$ as its rows, and $\mathbf{b} = \mathbf{As} + \mathbf{e} \mod q$ is the vector of b_i's. The vector $\mathbf{b}$ can be viewed as a lattice vector $\mathbf{As}$ (where the lattice is generated by the *columns* of $\mathbf{A}$, and $\mathbf{s}$ determines the integer linear combination of these lattice vectors) plus a small noise vector $\mathbf{e}$. Finding the closest lattice vector to $\mathbf{b}$ would imply finding $\mathbf{As}$, from which one can easily extract $\mathbf{s}$, and so solving the LWE problem is related to (a variant of) the CVP with target vector $\mathbf{t} = \mathbf{b}$ in this lattice modulo q.

Typically, in the context of lattice-based cryptography, we are concerned with discrete Gaussian error distributions $\chi = D_{\mathbb{Z},s}$ over the integers, where $\alpha := s/q \in (0, 1)$ is often called the (relative) error rate [58]. These are essentially continuous Gaussians with support restricted to the lattice. Importantly, the LWE problem is easily solved without the error term by applying Gaussian elimination. Otherwise it can be shown to be as hard as certain worst-case lattice problems [28].

Extensions of the LWE problem appear in the literature such as the binary-LWE problem [71], where the secret vectors are chosen uniformly from a binary set, rather than from $\mathbb{Z}_q^n$. This reduces the key size of $\mathbf{s}$, but comes at the cost of having to increase the parameters to guarantee security. Motivated by the fact that cryptographic schemes based on LWE are not efficient for practical applications due to the large size of $\mathbf{A}$ (resulting in key sizes of order n^2), Lyubashevsky *et al.* proposed cryptography based on the ring-LWE problem [61]. This is an algebraic variant of the LWE that operates over elements of polynomial rings instead of vectors. Intuitively, this modification adds additional structure to the sample matrix $\mathbf{A}$ so that it only requires storage of order n instead of n^2.

The LWE problem emerged as a basis for many cryptographic constructions such as public-key encryption schemes, identity-based encryption, and leakage-resilient encryption, among many others ([58], [89], [94], [95]).

III. ATTACKS: SOLVERS FOR LATTICE PROBLEMS

In this section, we identify the main existing attacks in the context of lattice-based cryptanalysis. Attacks, in this context, are algorithms that solve the various problems underpinning the security of lattice-based cryptosystems, which we enumerated and described in Section II.

A. LATTICE BASIS REDUCTION

Lattice basis reduction forms a cornerstone of lattice theory. In cryptanalysis, it is used to solve many of the problems mentioned in Section II. In fact, LLL, a seminal lattice basis

reduction algorithm, was used to break other cryptosystems even before lattice-based cryptosystems existed (e.g. [87]), and can still be used to attack certain lattice-based cryptosystems today.

LLL was the first algorithm for lattice basis reduction. Although originally described with rational arithmetic, the standard implementations today are LLL floating-point variants, which are considerably more efficient. These versions may incur errors, which are handled by the implementation itself. The most important variants of LLL include Schnorr and Euchner's floating point variant (LLL-SE) [102], and Nguyen and Stehle's variant (L^2) [81]. These variants are described in detail by their authors in a survey on LLL and its applications, published on the occasion of LLL's 25th birthday (see [83], [100], [105]). In 1994, Schnorr and Euchner presented another important variant of LLL, which uses a technique called "deep insertions" [102]. The literature indicates that this variant improves the quality of the basis substantially, while not increasing the running time of LLL too much; however, it is not as fast as LLL-SE in practice (see [105, pp 20–21], [110, Sec. 4.4.]).

Proposed by Schnorr in 1987 [98], the Block Korkine-Zolotarev (BKZ) algorithm is a generalization of LLL, and lays the foundation for the most efficient lattice basis reduction algorithms to date. BKZ offers a trade-off between the running time and the quality of the yielded basis, which is controlled by a parameter β. The algorithm solves the SVP on sliding windows (also called blocks) of β vectors, projected orthogonally to the span of the previous vectors in the basis [98]. Although in theory the SVP could be solved by any SVP-solver, enumeration-based SVP-solvers (see Section III-B) are usually used in this process. The shortest vector of the projected basis is then inserted into the original basis, and LLL is executed upon the whole basis so that linear dependencies are removed. The algorithm stops when a full set of sliding windows, from the beginning of the basis till the end, does not result in new vectors.

The most efficient lattice basis reduction algorithms today are variations of BKZ. These include BKZ 2.0 (proposed by Chen and Nguyen in 2011 [32]), progressive BKZ [14], and primal-dual BKZ [76].

B. THE SHORTEST VECTOR PROBLEM

The SVP has received considerable attention in lattice-based cryptanalysis over the last decades. The SVP is very important in this context as it is necessary to solve a relaxed version of the SVP, called α-SVP, to break specific cryptosystems. Most existing α-SVP-solvers do use SVP-solvers as part of their logic and thus the importance of understanding the SVP in practice. In particular, the BKZ algorithm referred in the previous section, can be used to solve the α-SVP, and it uses a SVP-solver (typically enumeration) as an essential building block.

There are various different SVP-solvers, but they can essentially be classified in two classes: enumeration-based solvers, running in superexponential time in the dimension but requiring almost no storage, and sieving-based solvers running in exponential time, but also requiring storage of size exponential in the lattice dimension.

Enumeration algorithms have been studied since the eighties, and progressively enhanced over time. In 2010, Gama *et al.* gave a comprehensive theoretical overview of different "pruning" strategies, including "extreme pruning", achieving significantly better results than with standard enumeration [42]. Pruning reduces the workload substantially, while at the same time the probability of success is reduced (but by a much smaller factor). The class of enumeration methods further includes the recent "discrete pruning" variant, based on Schnorr's Random Sampling Reduction method [13], [31], [41], [99].

Progress in sieving algorithms is considerably more recent. The first description of this method was only in 2001 [9], and the first truly practical implementations were released in 2008-2010 [74], [84]. There were no sieving algorithms capable of competing with enumeration until recently, with the appearance of HashSieve and LDSieve. The biggest downside of sieving algorithms is the large memory requirement, which becomes a serious bottleneck when solving the SVP in high lattice dimensions. Besides pure sieving algorithms, one might also include Voronoi cell approaches [4], [56], [73], [108] and discrete Gaussian sampling methods [2], [3] as sieving-based approaches, requiring exponential time and space, and following similar methodologies for solving the SVP.

Sieving algorithms have a few properties worth mentioning. First, they are asymptotically faster than enumeration ($2^{\mathcal{O}(n)}$ vs. $2^{\mathcal{O}(n \log n)}$), which means that they will be faster than enumeration in sufficiently high dimensions n. Second, some of these methods can take advantage of specific lattices, such as ideal lattices, whereas for enumeration algorithms it is not known how to exploit these ideal structures (see [97, Sec. 6.1]). Third, as convergent algorithms, sieving algorithms may be modified and still converge. This has in fact enabled many parallel variants to be designed over the last years, as we show in Section IV-B.

C. THE CLOSEST VECTOR PROBLEM

The CVP has been receiving less attention than the SVP. In particular, it was not until recently that the computational practicability and the scalability of CVP-solvers has been addressed. One of the reasons why the CVP has attracted less attention than the SVP may have to do with the fact that there is no public repository for the CVP. Furthermore, for certain methods it is known that they can solve SVP and CVP in practice in approximately the same amount of time, and so for these methods a CVP challenge would not be very useful.

There are two algorithms to solve the exact version of the CVP: the enumeration algorithm by Pohst, presented in 1981 [92], and an algorithm presented by Kannan [52], in 1983. Both extensions and improved versions of these algorithms have been published after that. In particular, in 1994,

Schnorr and Euchner proposed a significantly better version of Pohst's method [102], and recently, Ghasemmehdi and Agrell showed how one can eliminate some redundant operations in the algorithm [45].

There are also solvers that return an approximate solution of the CVP: the Nearest Plane algorithm, proposed by Babai [16], and certain sieving algorithms, such as the AKS algorithm [8]. The AKS algorithm, proposed by Ajtai *et al.*, is a randomized sieving algorithm to solve the SVP [9]. Later, it was extended to solve an approximated version of the CVP [8] (Blömer and Naewe proposed improvements for this algorithm [23]).

D. THE LEARNING WITH ERRORS PROBLEM

The literature suggests two different categories for algorithms solving this problem, namely algorithms solving the LWE problem directly (without targeting the underlying lattice) and solving the LWE problem via lattices (e.g. finding short/close vectors in a lattice). As for direct methods to solve the LWE problem, a naïve way is to generate enough LWE samples so that the secret $\mathbf{s}$ is recovered [93]. This algorithm requires both $2^{O(n \log n)}$ equations and running time. A similar algorithm uses a maximum likelihood method to solve the problem, thus requiring fewer samples ($O(n)$). However, it also runs in $2^{O(n \log n)}$ time [93].

The Blum-Kalai-Wasserman (BKW) [25] algorithm was published in 2003 as an attack on the Learning with Parity Noise (LPN) problem with an associated complexity of $2^{O(n)}$. It was generalized and later improved for the LWE problem [10], [11]. A different algorithm for solving the LWE problem was proposed in 2011 by Arora and Ge [15] and in their work it was demonstrated that by reducing the LWE problem to a system of (error-free) high-degree non-linear equations and solving this system, it is also possible to extract the desired solution. The complexity of this algorithm is $2^{O(n^{2\varepsilon})}$ which is exponential when $\varepsilon \geq \frac{1}{2}$. It is important to mention that both these approaches do not use lattice basis reduction (or lattices in general) to solve the LWE problem.

The distinguishing attack is one of the oldest attacks on LWE. It shows a direct relation between the LWE and α-SVP in certain lattices and can be used in combination with any α-SVP algorithm. Even though Lindner and Peikert claim the distinguishing attacked is outperformed by the decoding attack [58], recent performance estimates of sieving algorithms show that this attack could be the most promising on certain instances. However, since these arguments do not take memory consumption into account, they are somewhat debatable, but still show that the distinguishing attack should not be discarded.

The algorithms solving the LWE indirectly are extensively studied in the literature and can be characterized as attacks that use basis reduction such as Regev's [95] among many others [28], [32], [47]. To our knowledge, there is still no precise practical complexity analysis for large dimensions.

IV. THROUGHPUT PERFORMANCE AND SCALABILITY

This section gathers the implementations for the most important problems in lattice-based cryptography, while we comment on their performance, especially on parallel architectures.

A. LATTICE BASIS REDUCTION

There is substantial work regarding the practicability of lattice basis reduction algorithms, including LLL-SE, L^2, BKZ and BKZ 2.0.

1) LLL

In 2008, Backes and Wetzel published a shared-memory parallel LLL implementation, using pThreads, with moderate scalability [17]. In a follow-up paper, the same authors improved the results, achieving speedup factors of ≈3x and ≈4x for 4 and 8 threads respectively [18], as shown in Figure 5. This is, to our knowledge, the best result on parallelizing LLL so far, which means that, to this day, it has not been shown that LLL can scale linearly for high core counts.

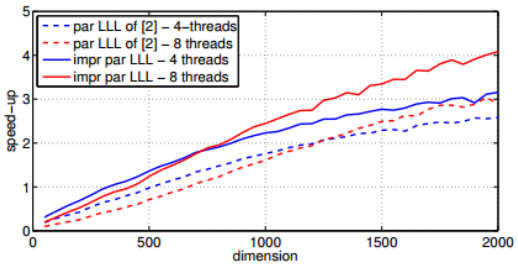

FIGURE 5. Speedup of a parallel LLL implementation presented in [18], on lattices up to dimension 2000 (reprinted from [18]). Note that "[2]" in the figure pertains to the second reference in [18], as the figure was directly reprinted from the paper.

Recently, Mariano *et al.* presented, for the first time, a vectorized LLL implementation, based on careful, cache-friendly data structure arrangements [66]. The authors started with a standard implementation, similar to that of the most popular LLL library, and made improvements to favour cache performance. They arrived at a key result: transforming the standard implementation to become cache-friendly only produced results after a certain lattice dimension. Before that lattice dimension, the additional overhead incurred by the cache-friendly implementation in accessing elements does not outweigh the performance boost given by increased locality of reference. The vectorized implementation (i.e. the cache-friendly implementation with two vectorized kernels) yielded better results than the standard implementation for any lattice dimension. In particular, by vectoring the dot product and the AddMul kernel, the authors achieved a 35% speedup.

The LLL implementation in NTL is algorithmically more efficient than the implementation used by Mariano *et al.*, thus being faster. However, the vectorization process suggested in [66], can improve NTL's implementation, as well as

the aforementioned parallel LLL implementation by Backes and Wetzel.

In 2016, Mariano conducted a series of tests comparing different LLL implementations (including NTL's, fpLLL's, plll and his own implementation) in his PhD thesis (see [63], Section 5.2.2). The experiments show the performance of the various implementations in terms of the quality of the basis and execution time, when executed with a single thread. fplll turned out to be faster than all other implementations, but the final basis was different in the overwhelming majority of tests (and thus so the quality). Mariano concluded that, in general, NTL and his own implementation tended to deliver better bases than fplll and the quality of the bases output by NTL and his implementation had small differences in terms of quality. A final important test in this section was to run an enumeration-based SVP-solver on the various resultant, LLL-reduced bases. Mariano concluded that, in general, the SVP-solver was faster on the bases output by his implementation (in comparison to fpLLL).

2) BKZ

Surprisingly, there are only a few papers on the parallelization of BKZ.

In 2014, Liu *et al.* presented a parallel BKZ variant implemented with MPI. Their variant is based on executing enumeration on several blocks in parallel, thus extracting parallelism at a coarse grain level. Whenever a new vector is found by the enumeration routine, it is inserted in the beginning of the block. After running the various blocks in parallel, LLL is executed over the entire basis, as in the original algorithmic description. The only metric Liu *et al.* used to judge the quality of the output bases was the length of the shortest vector in the basis, which is often smaller than that of the original algorithm.

In the same year, Arnreich and Correia presented AC_BKZ, another parallel BKZ variant, which is also based on running multiple calls of ENUM (the most popular enumeration-based SVP-solver) on different blocks, in parallel. Each ENUM call is sequential, and parallelism is extracted from running multiple ENUM calls in parallel. All the vectors found during this process are inserted in the basis, which is LLL-reduced before the subsequent round. The process is repeated until there are no new vectors resulting from ENUM calls. The main difference to Liu *et al.*'s version is that Arnreich and Correia ignore blocks with less than β vectors. The implementation scales well up to 4 threads, but stalls after that, as shown in Table 1. In addition, Correia concluded that the final bases are not as good as those yielded by the original BKZ algorithm [33].

In his PhD thesis, Mariano presented two different parallel implementations of the original BKZ algorithm (see Section 5.3.2 [63]). Essentially, both BKZ implementations used a parallel enumeration SVP-solver, implemented in two different models: task-based and demand-driven (SVP-solvers are covered in detail in Section IV-B). In the experiments, Mariano determined that the scalability of the

TABLE 1. Speedup (S) and Efficiency (E) of the AC_BKZ parallel implementation [33], for a lattice in dimension 60 for block-sizes 40, 45 and 50. Reprinted from [33].

	Block-size					
	40		45		50	
Threads	S	E	S	E	S	E
2	1.81x	91%	1.83x	93%	1.78x	89%
4	3.17x	79%	3.03x	76%	2.98x	75%
8	4.81x	60%	4.24x	53%	4.14x	52%
16	5.84x	37%	4.84x	75%	4.57x	29%
32	5.51x	17%	4.69x	15%	4.55x	14%

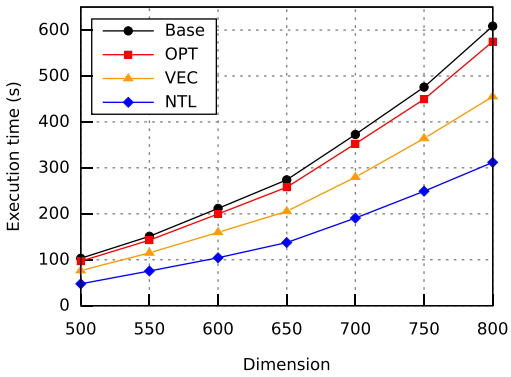

FIGURE 6. Execution time of the vectorized LLL implementations introduced in [66], and NTL's LLL implementation, on Ajtai lattices (reprinted from [66]). Base is the reference implementation, OPT is the reference implementation with cache-friendly data structures and VEC is the vectorized implementation.

BKZ implementations was similar to the scalability of the enumeration-based SVP-solver itself, as enumeration represents the majority of the computation in BKZ. The best scalability figures were achieved for higher block-sizes of BKZ, given that the scalability of enumeration is better for higher lattice dimensions (which is the block-size in BKZ). Increasing the block-size beyond 40, even for lattices in dimension 80, rendered BKZ impractical, as the complexity of BKZ grows exponentially with the block-size. The scalability of the demand-driven version was consistently better for high thread counts, and similar for the remaining cases. Table 2 shows the scalability of the demand-driven implementation (as Mariano concluded it is better than the task-based), for block-size 30.

A paper by Aono *et al.* from 2016, where the authors introduce progressive BKZ, presented a comparison of a few estimates (not actual experiments), including Lindner-Peikert's estimation [58], Chen-Nguyen's estimation [32] and their own estimation [14]. Aono's estimates indicate that his lower bound function offer much better single tread execution times than the state of the art algorithm, BKZ 2.0. There is no information regarding a possible parallel implementation of this variant. To the best of our knowledge, there are no results published on the performance and scalability of BKZ 2.0.

3) WRAP UP AND OPEN PROBLEMS

The parallelization of lattice basis reduction algorithms and LLL in particular has been studied, but to date there is no

TABLE 2. Parallel BKZ ($\beta = 40$) with demand-driven parallel enumeration, for 2-64 threads, on a 32-core + SMT machine. Reprinted from [63].

N	2T	4T	8T	16T	32T	64T
65	1.93	3.58	6.00	8.22	8.15	6.37
66	1.95	3.59	5.98	8.34	8.62	7.61
67	1.94	3.57	5.91	7.83	7.65	6.44
68	1.93	3.56	5.89	8.13	8.27	7.17
69	1.93	3.57	5.94	8.34	8.57	7.55
70	1.94	3.58	5.94	7.80	7.98	6.65
71	1.95	3.60	5.98	7.60	7.65	5.97
72	1.93	3.56	5.90	7.19	7.42	6.43
73	1.94	3.57	5.90	8.00	7.90	6.96
74	1.94	3.59	5.94	7.56	6.58	6.30
75	1.95	3.59	5.95	7.64	7.40	6.40

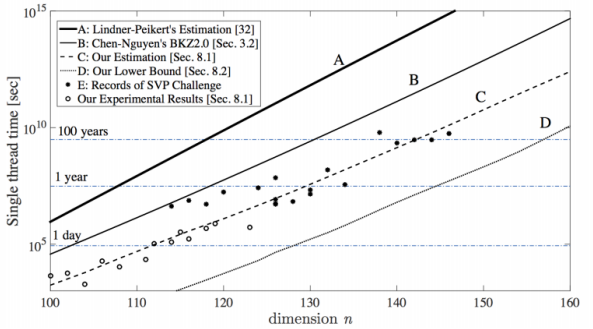

FIGURE 7. Comparison of various estimates, for a single thread execution time, of BKZ 2.0, and records on the SVP Challenge. Reprinted from [14].

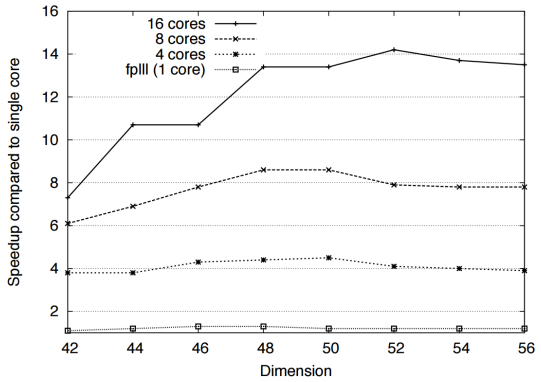

FIGURE 8. Scalability of Dagdelen *et al.* parallel ENUM implementations, for 1-16 threads. Reprinted from [37].

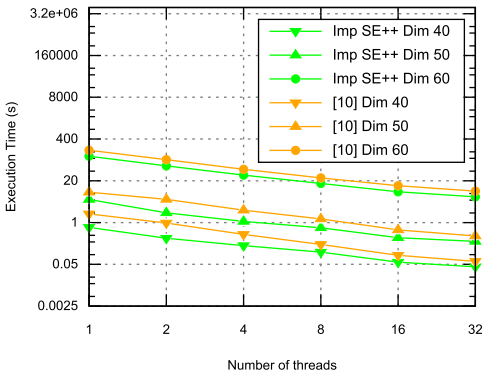

FIGURE 9. Throughput performance of Dagdelen *et al.* and Correia *et al.* enumeration-based SVP implementations, for 1-32 threads on a 16-core + SMT multi-core CPU machine. Reprinted from [33].

implementation for multi-core CPUs that scales linearly. The vectorization of some LLL kernels has been conducted with success, to which end it was shown that if the right data structures are used, significant gains can be obtained. The parallelization of BKZ has also been studied, and the best parallelization approach to date seems to be based on parallelizing the enumeration routine that underlies BKZ, as it consumes the bulk of the execution time of BKZ. To our knowledge, there are no papers on the parallelization of BKZ 2.0 to date. The parallelization of BKZ 2.0 can be done using the same demand-driven parallel implementation of ENUM presented in [63], according to the source. However, no practical experiments have been conducted to this day.

B. SVP

This section goes over the two main families of SVP-solvers - enumeration and sieving - pointing out the existent parallel implementations and commenting on their performance, based on the results available in the literature.

1) ENUMERATION

Enumeration algorithms have been studied thoroughly over the years, especially when compared to other classes of SVP-solvers. The first steps with enumeration-based algorithms date back to the 80s, when Pohst and Kannan presented seminal approaches, based on enumeration, to solve the SVP. Currently, the most practical full enumeration algorithms are ENUM [40] and SE++ [45], which differ mainly in the pre-processing. These algorithms can include pruning, a technique that discards computation at the cost of decreasing the probability of success (although by a much smaller degree).

There is a number of implementations of enumeration-based algorithms. Dagdelen *et al.* proposed the first parallel implementation of ENUM in 2010 [37]. This implementation scales well with the number of cores, attaining super-linear speedups in specific instances, as shown in Figure 8.

This implementation does not include any kind of pruning. In 2016, Correia *et al.* proposed a parallel version of SE++, which outperformed the implementation by Dagdelen *et al.* in terms of scalability and, more notably, in terms of throughput performance, with gains between 35% and 60% [35]. Figure 9 compares both implementations, when executing the SVP on two random lattices, on a 16-core + SMT machine. Correia also proposed the same parallelization approach to ENUM [33].

Last year, in his PhD thesis, Mariano presented two other implementations of ENUM, used in the BKZ implementation described in Section IV-A.2. The first implementation

TABLE 3. Scalability of the demand-driven and tasking-based ENUM implementations proposed by Mariano, for lattices in dimensions 30, 40, 50 and 60. Grayed-out cells represent better scalability. Reprinted from [63].

	Demand-driven ENUM				OpenMP-based ENUM			
Threads	30	40	50	60	30	40	50	60
2	1.65	2.00	2.01	1.97	1.20	1.91	2.01	1.92
4	2.34	3.92	4.05	3.94	1.72	3.93	3.86	3.83
8	2.27	6.41	8.14	7.74	1.84	6.82	7.43	7.82
16	1.90	9.65	16.01	15.06	1.10	11.56	15.42	15.19
32	1.84	10.19	28.79	25.81	0.51	16.59	29.20	28.88
64	1.28	10.24	37.93	39.00	0.06	1.06	38.63	38.54

is based on a demand-driven mechanism and one based on a tasking mechanism, which builds upon the work from Correia et al [35]. The results are shown in Table 3. Although the tasking-based mechanism scales relatively well, the demand-driven mechanism scales a bit better overall and is also more throughput-efficient in general. The main conclusion is that the demand-driven implementation allows certain optimizations that are not enabled by OpenMP in the tasking-based version. At the same time, Mariano reports that the demand-driven implementation's code is considerable more complex and error-prone, and was substantially harder to devise.

Another very important study important mentioning is the parallelization of extreme pruned enumeration. There are essentially two different studies in this matter, namely the parallelization by Correia in his MSc thesis [33] and then generalized in a journal paper [34]. This has been done with MPI, thus targeting distributed memory systems, and tested on shared-memory CPU-systems as well. The results are quite satisfiable given that the proposed implementation scales linearly for up to 4 processes. For 8 and 16 processes, the scalability decreases slightly, due to load imbalance. The authors show a direct correlation with the performance of BKZ, which ultimately determines the load of each of the running processes. Yet, the implementation achieves efficiency levels of over 90% for 4 or fewer processes and close to 90% for 8 processes, decreasing considerably thereafter.

2) WRAP UP AND OPEN PROBLEMS

The parallelization and throughput performance of enumeration-based SVP solvers are well studied. However, there are not, to the best of our knowledge, publicly available parallel implementations of BKZ 2.0, a considerably more efficient version of BKZ. Although Mariano has claimed that his demand-driven mechanism for parallelizing ENUM should work as well on pruned versions of enumeration, this claim is yet to be verified.

3) SIEVING

It was not until recently that sieving algorithms started to be regarded as practical and scalable algorithms. Up until 2010, enumeration-based algorithms were the most common SVP-solver used in practice.

In 2010, Micciancio and Voulgaris proposed ListSieve and GaussSieve [74], and the latter actually became the first sieving algorithm to surpass enumeration algorithms. There was especial interest around GaussSieve because sieving algorithms (and GaussSieve in particular) can take advantage of special lattice structures, such as ideals (in the form of ideal lattices). Motivated by the potential of GaussSieve, four particularly relevant GaussSieve implementations were proposed in the following years [27], [50], [69], [77].

- *Milde et al., 2011:* Milde and Schneider proposed the first parallel implementation of GaussSieve [77]. In brief, their implementation is based on a ring structure composed of several chunks of a global list, managed by each one of the running threads, whereon vectors float around. Because vectors may miss one another in the list, during the reduction process, many reductions are missed. As a result, the number of necessary iterations for convergence increases and the scalability decreases (this was shown to happen for more than 4 threads).
- *Ishiguro et al., 2014:* Ishiguro *et al.* presented a distributed-memory implementation of GaussSieve [50]. The list of vectors is not split among the running processes, as every process can access the global list. The implementation generates samples in batches (of a parameterizable size r) and synchronizes the processes between iterations. The authors claim that their implementation scales reasonably well with the number of processes, at the same time their implementation can also take advantage of ideal lattices. Unfortunately, the source code is not public and there are some important further details pertaining to synchronization that are not disclosed in the paper. There are some important downsides to this implementation, including the fact that good values for r must be determined empirically. Figure 10 shows the scalability of the implementation, presented in [50].
- *Mariano et al., 2014:* Mariano *et al.* proposed a parallel variant of GaussSieve that is based on a lock-free linked list (in particular an implementation of the Harris lock-free list [48]) [69]. The vectors in the system are stored in the lock-free list, which scales very well for most operations (and in particular better if the operations are not executed concurrently - which is the case in

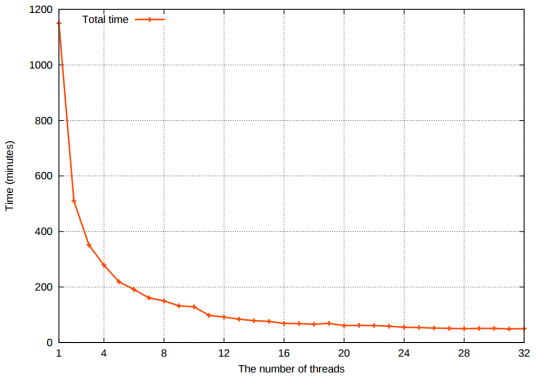

FIGURE 10. Scalability of Ishiguro *et al.* GaussSieve implementation, for 1-32 threads on a 16-core + SMT multi-core CPU machine. Reprinted from [50]).

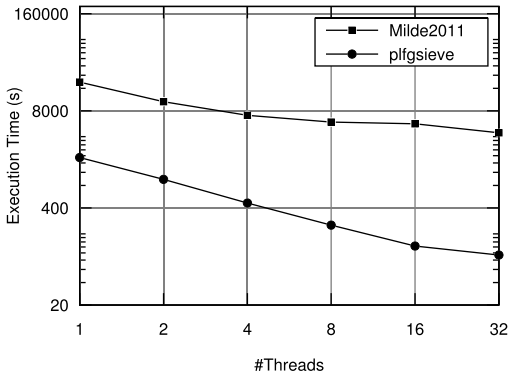

FIGURE 11. Scalability of Mariano *et al.*'s and Milde and Scheider's GaussSieve implementations, for 1-32 threads on a 16-core + SMT multi-core CPU machine. Reprinted from [69]).

GaussSieve). This version also relaxes the properties of the algorithm a little bit, as it cannot ensure that every vector can see every other vector in the system. The implementation was shown to scale very well for any number threads on a 32-core CPU system. Figure 11 shows the scalability of this implementation, presented in [69], in comparison to that of Milde and Schneider.

- *Bos et al, 2014:* Bos *et al.* proposed a different approach to parallelize GaussSieve, for distributed systems, which includes a routine to exploit ideal lattices [27]. Put simply, the parallelization consists in replicating the same vector across different processes, which reduce it against their own chunk of the distributed list. Vectors are generated in batches and reduced against the individual lists, by each process. The surviving vectors are pairwise reduced and added to the list of one process at the end of each round. The vectors that do not survive are moved to a different list, where eventually only the shorter - or minimal representative - of each vector is kept. This is joint to a global stack and used as the vectors of the following iteration. This implementation has various synchronization at various points, including broadcasting samples, agreeing on minimal representatives, and concatenating lists. The efficiency of the implementation is reported in Table 4, retrieved from the original paper.

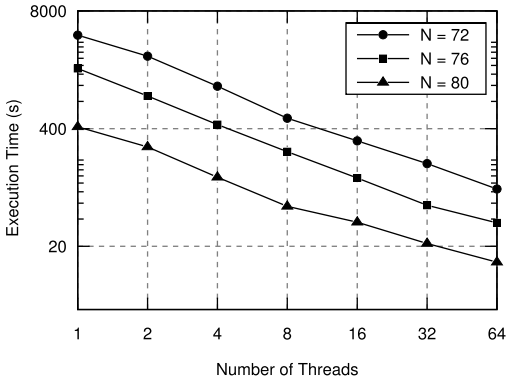

FIGURE 12. Scalability of Mariano *et al.*'s LDSieve implementation, for 1-64 threads on a 32-core + SMT multi-core CPU machine. Reprinted from [62]).

Although GaussSieve was a very important mark in the development of sieving algorithms, enumeration with extreme pruning was proposed in the same year. As a result, enumeration remained as the best SVP-solver for random lattices in the years that followed.

Also in 2014, Becker *et al.* proposed a sieving algorithm based on overlattices, called Overlattice sieve [19]. The authors presented a scalable implementation along with the introduction of the algorithm. Although no scalability figures are presented, the authors argue that their algorithm is "*suitable for SIMD implementation, not only multi-threading*" (sec. 4.5) and "highly parallelizable" (sec. 4.9).

In 2015, Laarhoven proposed HashSieve [55], putting sieving algorithms back on the map. HashSieve was shown to be competitive with enumeration with extreme pruning. In the same year, Mariano *et al.* proposed a parallel implementation of HashSieve, which slightly relaxes the properties of the algorithm and was shown to scale well on CPU system with up to 64 threads [67], [68]. Later on, on a follow-up paper, it was shown that with specific improvements, the parallel variant of HashSieve could in fact scale linearly or almost linearly on high thread count CPU systems. In addition, several memory access improvements were proposed, including prefetching techniques and the use of data mapping policies [64], [67].

Earlier this year, Mariano *et al.* proposed a parallel implementation of LDSieve that scales almost linearly on multi-core CPUs with up to 64 threads, with linear speedups up to 16 threads and almost linear for higher thread counts [62]. The implementation uses the underlying probable lock-free mechanism used to parallelize HashSieve in [67], although applied to the single hash table in the system. The implementation is also optimized at various levels, including vectorized routines for vector reduction and memory efficient access patterns [67]. Figure 12 shows the scalability of the implementation.

Earlier this year, Yang *et al.* proposed a GPU implementation of GaussSieve for ideal lattices, which builds

TABLE 4. Scalability of Bos *et al.* GaussSieve implementation, for 1-256 cores (data taken from [27]).

C	80				88				96			
	t_{GH}	t_{Coll}	S	E	t_{GH}	t_{Coll}	S	E	t_{GH}	t_{Coll}	S	E
8	1918	3010	–	–	28961	45106	–	–	506841	654483	–	–
16	883	1592	1.9	.95	16794	23994	1.9	.95	252246	336288	2.0	1.0
32	533	820	3.7	.93	10445	14176	3.2	.80	125907	167790	3.9	.98
64	248	455	6.6	.83	4259	6037	7.5	.94	75676	101303	6.5	.81
96					3638	4984	9.1	.76	46774	70740	9.3	.77
128					2292	3392	13.2	.83	41196	54907	11.9	.74
160					2267	3206	14.1	.71	27864	36048	18.2	.91
192					2276	3104	14.5	.60	27038	35179	18.6	.78
224					1960	2849	15.8	.56	24873	33115	19.8	.71
256					1980	2726	16.5	.51	23442	30374	21.5	.67

upon Ishiguro's and Bos's work [109]. The authors argue that their implementation works on multi-GPU platforms, achieving parallel efficiencies of 45-86%, depending on the lattice dimension and the number of used GPUs. The authors used this implementation to find a short vector on a 130-dimensional ideal lattice, a computation that required 824h on 8 GPUs.

4) WRAP UP AND OPEN PROBLEMS

The parallelization of sieving algorithms has been extensively studied, with good results. There are a few open problems including 1) whether the latest advances in sieving algorithms can be adapted to take advantage of ideal lattices, and 2) whether these more advanced sieving algorithms, such as HashSieve or LDSieve, can effectively be ported to GPUs.

C. CVP

Unlike the SVP, the CVP has not got much attention in the past years in terms of practical assessments. A sequential implementation of an enumeration-based CVP-solver can be found in Magma.[5] However, it requires users to contribute to distribution costs (licenses start at 1000 euros). fplll includes a non-supported implementation of the Nearest Planes algorithm for the approximate CVP problem, and an enumeration-based CVP-solver.

Correia *et al.* proposed in [35] a parallel implementation of the SE++, an improved version of the Schnorr-Euchner enumeration, which was proposed by Ghasemmehdi and Agrell [45]. The parallel SE++ scales linearly for up to 8 threads and almost linearly for 16 threads. The implementation can also benefit from the SMT technology, since the dependencies between the instructions prevent the full use of the functional units within each core. Figure 13 shows execution time of the implementation, running with 1-32 threads, with BKZ-reduced bases.

To the best of our knowledge, for the CVP, there are neither publicly available implementations for the Kannan algorithm nor for CVP-solvers based on sieving.

[5]http://magma.maths.usyd.edu.au/magma/

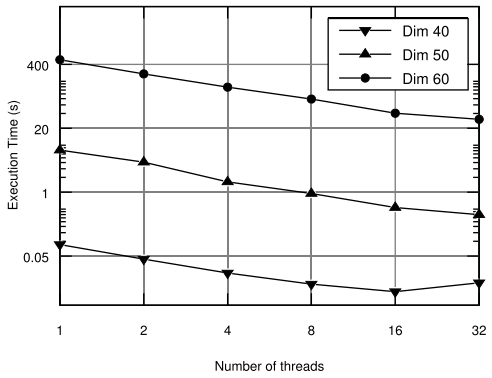

FIGURE 13. Scalability of Correia *et al.*'s SE++ implementation, for 1-32 threads on a 16-core + SMT multi-core CPU machine. Reprinted from [35].

1) WRAP UP AND OPEN PROBLEMS

Correia *et al.*'s work showed that enumeration-based CVP-solvers can scale linearly on multi-core CPUs. However, it is still unclear if sieving-based CVP-solvers can also scale well, and what optimization opportunities can in fact be carried out. Finally, the scalability of CVP-solvers have not been tested on massively parallel architectures such as GPUs.

D. LWE

Similarly to what happens with the CVP, only few parallel implementations of algorithms that solve the LWE problem were proposed to date. The first parallel implementation solving this problem dates back to 2014, when Bischof *et al.* [22] assessed the performance and scalability of Lindner and Peikert's "decoding attack" [58]. This algorithm converts an instance of the LWE problem to an instance of the CVP. Bischof *et al.*'s implementation achieves speedup factors of more than 11x on a machine with four CPU chips, totaling 16 cores.

In 2016, Kirshanova *et al.* developed a parallel implementation of the same algorithm [53]. It differs from the former as it uses Liu and Nguyen's pruned enumeration (see [59]) instead of the nearest planes algorithm. The authors compared this implementation to their own implementation of the

parallel decoding attack with the nearest planes algorithm. They showed that the version with the pruned enumeration runs significantly faster than the version with the nearest planes algorithm, while achieving almost linear scalability.

1) WRAP UP AND OPEN PROBLEMS

The literature is short in implementations of algorithms for the LWE problem. The known implementations, developed by Bischof *et al.* [22] and Kirshanova *et al.* [53], show that algorithms for the LWE can be scalable.

V. TOOLS FOR LATTICE-BASED CRYPTANALYSIS

Other than the spare implementations enumerated in Sections III and IV, there are some libraries and software available online that can be used for practical lattice-based cryptanalysis. In the following, we identify the main libraries that contain implementations of attacks, while enumerating those attacks.

A. fpLLL

Currently, fplll is arguably the most important open-source library available for lattice cryptanalysis. It is written in C++, and provides efficient implementations of floating-point LLL variants [78], [81], BKZ, and various SVP solvers based on enumeration and sieving. For the BKZ, three variants are available, namely the classical BKZ, the Slide Reduction and the Self-Dual BKZ variants [43], [76]. Some of the most recent updates also include implementations of the BKZ 2.0, which is one of the fastest lattice basis reduction algorithms to date, and the GaussSieve, a sieving-based SVP-solver. To the best of our knowledge, the only implementation of BKZ 2.0 that is publicly available is contained in fplll. The library also provides an implementation for the CVP, with an enumeration based algorithm.

B. NTL

Before fplll was released, NTL was the main open library for lattice algorithms, as well as many other number-theoretic functions. It is written in C++ and includes several lattice algorithms, such as LLL and BKZ. These algorithms can be run either with the Gram-Schmidt orthogonalization process or Givens rotations. All algorithms are implemented with several precisions, which allows one to trade performance for output reliability. Higher precision implies that the chance for errors, which may change the output, is lower, at the cost of performance. In the context of lattice-based cryptanalysis, NTL is still one of the most used libraries in practice, due to its fast implementations of LLL and BKZ. Some of the best (closed-source) algorithms to date were implemented based on NTL, such as BKZ 2.0 and progressive BKZ [14], [32].

C. pLLL

Besides fplll and NTL, plll is one of the most complete lattice libraries available. plll is written in C++ and contains several implementations of LLL, BKZ and SVP-solvers. The available LLL implementations include the classic LLL, the unprojected LLL and the Siegel LLL. The classic LLL uses the original Lovász condition [57], which compares the projected lengths of two adjacent vectors. Vectors are projected onto the orthogonal complement of all previous vectors. The unprojected LLL uses a simpler condition that compares the unprojected lengths of two adjacent basis vectors. This is essentially an attempt to sort the basis vectors by increasing norm, while size-reducing the basis. The Siegel LLL uses the Siegel condition, which allows to prove the same bounds on the output quality as the Lovász condition. The classic LLL implementation allows both regular size reduction and deep insertions (in various modes and with several options).

There is also a number of BKZ implementations available in plll, including the Schnorr-Euchner BKZ [102], the simplified BKZ, the terminating BKZ, the Primal-Dual BKZ and the Slide Reduction, among others. The simplified BKZ and the terminating BKZ were proposed by Hanrot *et al.* in [47]. The Primal-Dual BKZ implementation in plll is a mixture of slightly different BKZ descriptions, including H. Koy and Schnorr's descriptions (see plll's documentation[6] for more). The Slide Reduction algorithm was proposed by Micciancio and Walter in [75].

plll implements a few SVP-solvers too. Regarding enumeration-based solvers, plll implements an improved variant of the Kannan-Schnorr-Euchner enumeration - including a parallel implementation - and a single threaded version of Schnorr's new SVP solver, described in [101]. There are also versions of a few sieving algorithms available, such as the List Sieve, List Sieve Birthday and the Gauss Sieve.

Finally, there is also an implementation of the Voronoi cell algorithm for the SVP. However, the standard Voronoi cell algorithm is not competitive with enumeration and sieving in any reasonable dimension (say $n \geq 30$).

D. OTHER LIBRARIES AND SOFTWARE

1) LatEnum

LatEnum is a small C++ library that provides enumeration-based SVP- and CVP-solvers. It also contains a parallel MPI implementation of enumeration for distributed memory systems.

2) THE LATTICES PACKAGE FOR HASKELL

The Lattices package for Haskell[7] includes an implementation of the Babai nearest planes algorithm [16], a "very basic" LLL implementation with exact arithmetic [57] and a floating point LLL implementation, based on Schnorr and Euchner's algorithm, proposed in [102].

3) LATTICE BASIS REDUCTION IN MATHEMATICA

The "extended lattice reduce algorithm" package[8] adds an extended LLL function to the builtin Mathematica lattice reduction function. This function computes a reduced basis,

[6] https://felix.fontein.de/plll/docs/latticereduction.html
[7] https://hackage.haskell.org/package/Lattices
[8] http://library.wolfram.com/infocenter/MathSource/681/

and computes a transformation that relates the reduced basis with the original list of generators. The documentation does not describe which basis reduction algorithm it implements, but practical tests suggest it is a slightly stronger variant of LLL.

4) LLLBases PACKAGE FOR MACAULAY2 LIBRARY

Macaulay2 is an algebraic geometry and commutative algebra software system. The LLLBases package extends this software system by implementing several variants of LLL. Some of these are implemented in the Macaulay2 engine, while others by NTL. The algorithms were implemented to work for different precisions. In addition, LLL with Givens rotations, instead of Gram-Schmidt and BKZ are also available.

5) ELEMENTAL

Elemental is a C++ library for distributed-memory dense and sparse-direct linear algebra, conic optimization, and lattice reduction. The library contains an extension of Householder-based LLL to real and complex linearly dependent bases and generalizations of BKZ 2.0 to complex bases incorporating "y-sparse" enumeration.

6) FLINT

FLINT is a number theory library, written in C. It includes an optimised LLL implementation and a large quadratic sieve for factorization of integers. Currently, these implementations are only available in versions 1.x, but they are planned to be ported to versions 2.x in the future.

7) LiDIA

LiDIA is a C++ library for computational number theory that contains an implementation of LLL.

8) LLLplus

LLLplus is a lattice reduction package, developed in Julia. It contains an implementation of the LLL lattice reduction, the Seysen lattice reduction and an enumeration-based CVP-solver, which can also be used to solve the SVP.

Figure 14 summarizes the various libraries and packages and the implementations of attacks for lattice-based cryptanalysis they provide.

VI. CHALLENGES FOR LATTICE CRYPTANALYSIS

Besides the libraries supporting implementations of various algorithms used in lattice cryptanalysis mentioned in the previous section, allowing anyone to test these algorithms in practice, another useful tool in lattice cryptanalysis is having public challenges for people to solve. These may be concrete instantiations of cryptosystems, asking people to recover the secret keys, or randomly chosen lattices and lattice problems for anyone to solve. These have greatly aided in providing anyone an idea of what is feasible/infeasible with current attacks, and stimulated work on fast implementations of these algorithms, to achieve higher records.

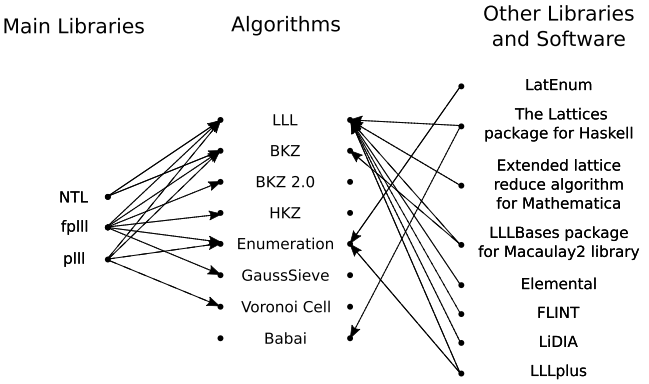

FIGURE 14. Libraries, packages and available implementations of attacks for lattice-based cryptanalysis.

A. GGH CHALLENGES

Probably the first public challenges for lattice-based cryptography were the GGH challenges [46], which can still be found online,[9] and are based on solving certain CVP instances in GGH lattices. Explicit instantiations of the cryptosystem are given, and the public is asked to recover the secrets. Although the authors initially hoped that the challenges in dimensions 300 and higher would be impossible to solve, two years later Nguyen showed [110] that the challenges up to dimension 350 can be solved with relative ease. Partly due to the inefficiency of the GGH cryptosystem in high dimensions, and more efficient lattice-based cryptosystems being designed in later years, people stopped working on breaking the last unsolved GGH challenge.

B. NTRU CHALLENGE

The NTRU cryptosystem [49], closely related to cryptography based on the ring-LWE problem, is the first and to date perhaps the only lattice-based cryptosystem which was commercialized into a product. To evaluate the practical security of NTRU, as well as show potential customers that indeed NTRU cannot be broken by the world's leading experts, they created the NTRU challenge,[10] where system parameters and public keys are provided, and the challenge is to recover the private key. These challenges came in various difficulty levels (lattice dimensions), with cash prizes of between $1000-$5000 for anyone who first solves any of these challenges.

To date, seven challenges (out of 27) have been successfully solved, using a combination of BKZ basis reduction, fast enumeration techniques, and the so-called "hybrid attack" specifically aimed at NTRU and NTRU-like lattices.

C. HIMMO CHALLENGE

Very recently, Philips designed a new key predistribution scheme called HIMMO [44], for which it was claimed that the strongest attacks would be based on solving certain lattice problems underlying this cryptographic primitive. Similar to

[9] http://groups.csail.mit.edu/cis/lattice/challenge.html
[10] https://www.onboardsecurity.com/products/ntru-crypto/ntru-challenge

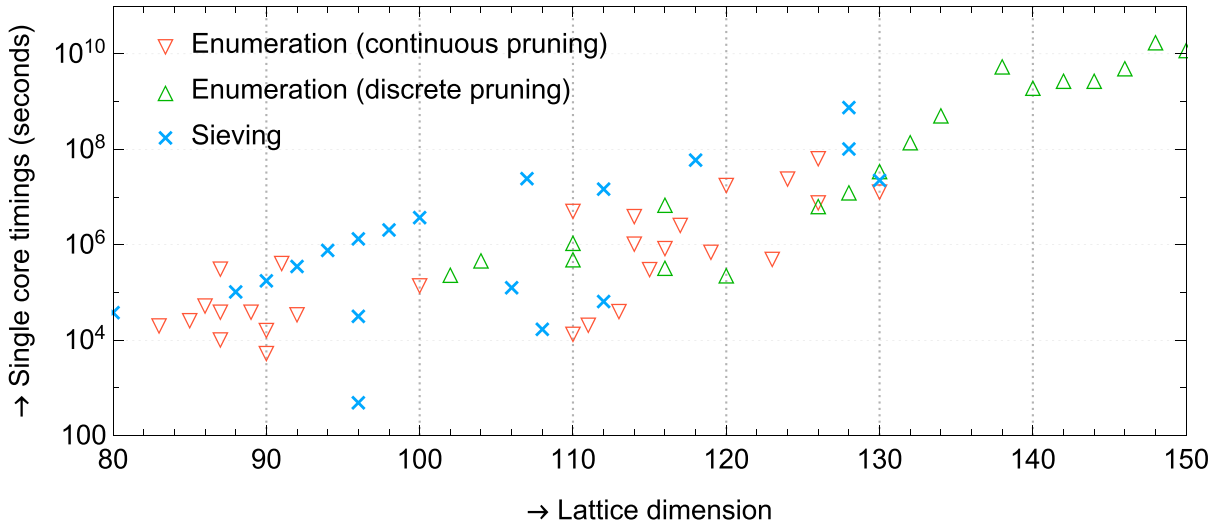

FIGURE 15. Single-core timings for the SVP challenge records, as provided by the authors in the comments on the records of the SVP challenge, or in associated scientific publications. The highest records consumed up to 10^{10} single-core seconds, corresponding to over 150 days of computation on a 1000+ core cluster.

NTRU, to obtain more confidence in their scheme before commercializing it they set up the HIMMO challenge,[11] challenging anyone to recover secrets from randomly chosen public keys. There were also several cash prizes available to anyone who first solved these challenges.

In the 2015 edition of this challenge, some of the separate HI and MMO challenges were solved, and after a vulnerability in the scheme was found, all HIMMO challenges were solved. The HIMMO scheme was later fixed, and new challenges were constructed, where the same attack would not work. However, in late 2016 researchers at Philips discovered another vulnerability in their scheme [96]. To date this revised HIMMO scheme has not been fixed, and now only claims to possess ''limited collusion resistance'' rather than full, post-quantum collusion-resistance.

D. LATTICE CHALLENGE

In the class of challenges aimed concretely at underlying lattice problems, rather than real cryptosystems based on lattices, the first well-known challenge website is the lattice challenge[12] [30]. Here lattices related to the worst-case/average-case reduction of Ajtai are constructed, and researchers are challenged to find sufficiently short vectors in these lattices. In practice this mostly translated to a ''lattice basis reduction challenge'', as the best methods to solve these challenges were based on a combination of strong lattice basis reduction with enumeration in a small sublattice of the entire lattice.

Besides two new records from January 2017, no new records were found in this database since 2013. The highest records attained were in dimension around 800, using a combination of BKZ and enumeration.

E. SVP CHALLENGE

In contrast to the lattice challenge, where only a reasonably long approximate shortest vector was needed to solve a challenge, the SVP challenge[13] focused on pure SVP solvers, by asking for (almost) exact solutions to the shortest vector problem in randomly generated lattices. This challenge has received and still receives the most attention from the community, with serious efforts being made to solve the highest records.

At this point, the highest records are in dimension around 150, using up to hundreds of core years on giant clusters, and are based on enumeration with discrete pruning [13]. An overview of the records from the SVP challenge is given in Figure 15. In total, this database contains over 350 entries, with the highest solved challenge progressing from dimension 110 in 2010 to dimension 150 in 2017.

F. IDEAL SVP CHALLENGE

With the cryptographic community focusing more and more on cryptographic primitives based on ideal lattices, due to their better efficiency, also an ideal SVP challenge[14] was soon launched [91]. Here the randomly chosen lattices are ideal lattices, and again only exact or almost exact solutions to the SVP suffice to solve the challenges. This challenge has received somewhat less attention than the standard SVP challenge, due to most methods not being able to exploit the ideal structure of ideal lattices to obtain better records.

The current highest records in this database are based on lattice sieving methods which do exploit the ideal structure [20], [27], [50], [109], and standard enumeration methods. These implementations however have not been

[11] https://www.himmo-scheme.com/challenge/
[12] https://www.latticechallenge.org/
[13] https://www.latticechallenge.org/svp-challenge/
[14] https://www.latticechallenge.org/ideallattice-challenge/

optimized and run on as large clusters as some of the SVP challenge records, and so the highest solved challenges are only in dimension around 130.

G. LWE CHALLENGE

With LWE also playing an increasingly prominent role in efficient lattice-based cryptography, an LWE challenge[15] was also created [29]. Due to the more complex parameter selection in LWE, a wide spectrum of challenges was created here, which may assist in learning which parameter choices to avoid. Being a fairly new challenge, only launched in 2016, not that many challenges have been solved to date.

H. RING-LWE CHALLENGE

Similar to the ideal SVP challenge variant of the SVP challenge, a ring-LWE challenge[16] has also been constructed [36] to assess the practical feasibility of solving the ring-LWE problem in low/moderate dimensions. These challenges target solving the ring-variant of the LWE problem, commonly used in cryptography due to the smaller related key sizes. Having been introduced only very recently, none of the ring-LWE challenges have been solved at the time of writing.

VII. WHAT THE FUTURE HOLDS

In the last years, there has been a considerably body of work contributing to improving the tools and processes of lattice-based cryptanalysis. In particular, considerable effort was put into understanding the potential of attacks against lattice-based cryptosystems, and improving them. For instance, in 2010, sieving-based SVP solvers were only able to reach dimension 66 [73], while in 2016 they reached dimension 107 [67]. Enumeration-based solvers have also evolved considerably, especially with the proposal of the extreme pruning technique [42], which allowed experiments up to dimension 120. These and many other experiments showed that certain architectures can be used effectively to implement SVP-solvers. For instance, sieving algorithms scale well on multi-core CPUs e.g. [64], [69] and enumeration algorithms can be effectively offload onto CPUs and GPUs [54].

We think it is important to talk about the limitations found throughout the scrutiny of these and other attacks, which we covered earlier in this paper. In the following, we address these limitations, considering some predictable advances on computer architecture. We also address computer architectures that have not been used for lattice-based cryptanalysis yet.

A. MEMORY AND COMPUTATION

The experiments with SVP-solvers gathered in this survey suggest that sieving algorithms are memory-bound and enumeration algorithms are compute-bound. In today's computer architectures, memory is less efficient than computation [80], thus penalizing sieving-based algorithms with regard to enumeration. The current direction for large computational systems seems to lead to a decrease in the memory per core [51], [79], which will not favour sieving algorithms as they have high memory requirements.

Currently, single GPU models are not practical to implement sieving, given that GPU cards only have up to 24/32 GB of memory [1], [85], [86], depending on the manufacturer. The current roadmaps provided by GPU manufacturers do not indicate how much memory exactly we can expect in the next generations of GPUs. Either way, as current practical dimensions for the latest sieving algorithms require upwards to one TB of RAM [65], it is unlikely that we can use GPUs exclusively to run sieving on high dimensions. In addition, an inherently hard task associated with this is the partition of data for the latest sieving algorithms, such as HashSieve, and the impact of data partition in throughput performance and the convergence rate of the algorithm. In CPU+GPU models, there is the possibility to extend the available memory with technology such as NVIDIA's Unified Memory technology, which permits to use the host's memory for computation executed by the GPU. However, there is a latency penalty incurred by this technology, which makes it unclear whether it would pay off to use it for sieving algorithms.

B. ENERGY CONSUMPTION

We also argue that energy consumption is a wall that can be hit by an adversary in practice. An interesting point would be to determine how big a penalty in throughput would be if we were to limit the energy consumption. This is to say that an adversary hits the energy consumption wall, and we are interested in finding out whether the penalty in execution time / throughput is enough to compromise the attack. In other words, we can define this problem as finding whether the decrease or limitation in energy consumption would translate into an execution time increase that is favourable, i.e. proportionally bigger, or not. As FPGAs offer the possibility to trade off execution time for energy consumption, by adjusting the clock frequency, we could use FPGAs to replicate this problem at a smaller scale. Either way, this topic deserves study that goes beyond FPGAs and reconfigurable hardware, so we point out some future lines of work that build upon this idea, in the next sub section.

C. RECONFIGURABLE HARDWARE

Reconfigurable hardware has in fact been less explored in the context of lattice-based cryptanalysis. To our knowledge, the study that relates reconfigurable hardware to lattice-based cryptanalysis is limited to the use of an FPGA to accelerate lattice reduction [38], a study that showed promising results. A big problem related to the usage of FPGAs pertains to the synthesizing time and the RTL designing skills. In [12], the authors proposed a framework for error-correcting codes (in particular for low-density parity-check (LDPC) decoders) that solves this problem for that specific context. A similar framework could be designed for the context of lattice-based cryptanalysis, allowing C/C++ programmers to conduct

[15] https://www.latticechallenge.org/lwe_challenge/
[16] http://web.eecs.umich.edu/cpeikert/rlwe-challenges/

further experiments in this realm. In fact, high-level synthesis in general is going in this direction [24]. However, we point out that regardless of the improvements on the programmability of FPGAs, memory is still a limited resource in these devices, either in terms of availability or technology (access time / speed).

We are walking toward a generation of devices that include reconfigurable and non-reconfigurable hardware physically close to each other, e.g. [39]. A big step in this direction was the acquisition of Altera by Intel, in 2015.[17] In this setup, we can envision that the programmable part of the die takes care of specific operations that are not very efficient on the CPU/non-reconfigurable component. It would be interesting to account for this today, using reconfigurable hardware as a co-processor to speed up the kernels of the current attacks which CPUs are not particularly effective at solving.

D. ALGORITHMIC ADVANCES

Besides improvements on the practical side of implementations of existing lattice algorithms, the future will surely bring new theoretical advances in this area as well. For instance, from the 1980s up until 2001 enumeration was the only known method to solve SVP. Then, in 2001 sieving was invented, which initially appeared completely impractical, but recently turned out to be quite competitive when making certain heuristic modifications. In the meantime various modifications to enumeration were made as well, theoretically improving upon the initial enumeration methods by huge factors.

Although it is hard to predict the theoretical advances in this area, the future will undoubtedly present new ideas and new techniques, which either help further improve existing methods in theory and in practice, or suggest completely new approaches for solving lattice problems. Regardless of this progress on the algorithmic side, our ever-increasing experience with existing methods and techniques will assist in estimating their practicability faster, when they arrive. On the theoretical side, there have been quite some recent advances on faster quantum algorithms for ideal lattice-based cryptography, and a breakthrough like Shor's may not be far away. If certain classes of lattice problems turn out to be easy to solve on quantum computers, the focus of the community will shift away from these primitives, which will further motivate a practical assessment of the hardness of the remaining lattice problems.

1) FUTURE LINES OF RESEARCH

While enumeration has been effectively ported to CPU+GPU platforms, it is still unclear whether modern sieving algorithms, such as HashSieve and LDSieve, can also be offload to such platforms, and to GPU-only platforms. We believe that this would be an important task because as we said before, current sieving algorithms hit a memory wall before dimension 110 [65], and it would be interesting to understand the behaviour of sieving for higher dimensions. Adapting the current parallel implementations of the best sieving algorithms to take advantage of ideal lattices should also be a high priority line of research, as understanding whether HashSieve and LDSieve continue to scale well on ideal lattices is of major importance.

Regarding lattice-reduction algorithms, it would be very interesting to test other SVP-solvers, such as sieving, within BKZ. There are a few reasons to do this. One, sieving is shown to scale better than pruned enumeration on multi-cores, so the sliding windows on BKZ could be run with sieving, and scale well with the number of cores in the system. Note that the best way to parallelize BKZ is by parallelizing these windows, as they represent the vast majority of the execution time for relatively sized lattices. Two, sieving could also be adapted to discard computation, in a similar way to what pruning does in enumeration, and thus have a close algorithm to BKZ 2.0. Three, sieving can take advantage of ideal lattices, which means that BKZ could actually take advantage of ideal lattices (at least partially as the internal mechanism of BKZ may actually destroy the ideal properties).

Finally, given that energy consumption may become a bottleneck for a real-world adversary, it would be very interesting to assess the suitability of low-power parallel architectures, such as multi-core ARM processors and DSPs, for attacks such as sieving and enumeration.

[17] https://newsroom.intel.com/news-releases/intel-completes-acquisition-of-altera/

ACKNOWLEDGMENTS

Artur Mariano thanks HiPEAC network for a collaboration grant awarded to visit Gabriel Falcão in late 2016, when the write up of this paper started.

REFERENCES

[1] *AMD FirePro S9170 Server GPU*, Adv. Micro Devices, Sunnyvale, CA, USA, 2016.

[2] D. Aggarwal, D. Dadush, O. Regev, and N. Stephens-Davidowitz, "Solving the shortest vector problem in 2^n time via discrete Gaussian sampling," in *Proc. STOC*, 2015, pp. 733–742.

[3] D. Aggarwal and O. Regev. (Aug. 2013). "A note on discrete Gaussian combinations of lattice vectors." [Online]. Available: http://arxiv.org/abs/1308.2405

[4] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[5] M. Ajtai, "Generating hard instances of lattice problems (extended abstract)," in *Proc. STOC*, New York, NY, USA, 1996, pp. 99–108.

[6] M. Ajtai, "The shortest vector problem in L_2 is NP-hard for randomized reductions," in *Proc. STOC*, 1998, pp. 10–19.

[7] M. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," in *Proc. 29th Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, 1997, pp. 284–293. [Online]. Available: http://doi.acm.org/10.1145/258533.258604

[8] M. Ajtai, R. Kumar, and D. Sivakumar, "Sampling short lattice vectors and the closest lattice vector problem," in *Proc. 17th IEEE Annu. Conf. Comput. Complex.*, May 2002, pp. 53–57.

[9] M. Ajtai, R. Kumar, and D. Sivakumar, "A sieve algorithm for the shortest lattice vector problem," in *Proc. 23rd Annu. ACM Symp. Theory Comput.*, 2001, pp. 601–610.

[10] M. R. Albrecht, C. Cid, J.-C. Faugère, R. Fitzpatrick, and L. Perret, "On the complexity of the BKW algorithm on LWE," *Des., Codes Cryptogr.*, vol. 74, no. 2, pp. 325–354, Feb. 2015.

[11] M. R. Albrecht, J.-C. Faugère, R. Fitzpatrick, and L. Perret, *Lazy Modulus Switching for the BKW Algorithm on LWE*. Berlin, Germany: Springer, 2014, pp. 429–445.
[12] J. Andrade *et al.*, "From low-architectural expertise up to high-throughput non-binary LDPC decoders: Optimization guidelines using high-level synthesis," in *Proc. 25th Int. Conf. Field Program. Logic Appl. (FPL)*, Sep. 2015, pp. 1–8.
[13] Y. Aono and P. Q. Nguyen, "Random sampling revisited: Lattice enumeration with discrete pruning," Cryptol. ePrint Arch. 2017/155, 2017. [Online]. Available: https://eprint.iacr.org/
[14] Y. Aono, Y. Wang, T. Hayashi, and T. Takagi, "Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator," in *Proc. 35th Annu. Int. Conf. Theory Appl. Cryptograp. Techn.*, vol. 9665. Vienna, Austria, May 2016, pp. 789–819. [Online]. Available: https://doi.org/10.1007/978-3-662-49890-3_30, doi: 10.1007/978-3-662-49890-3_30.
[15] S. Arora and R. Ge, "New algorithms for learning in presence of errors," in *Proc. 38th Int. Colloq. Conf. Automata, Lang. Program. (ICALP)*, Berlin, Germany, 2011, pp. 403–415.
[16] L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, Mar. 1986.
[17] W. Backes and S. Wetzel, "A parallel LLL using POSIX threads," Dept. Comput. Sci., Stevens Inst. Technol., Hoboken, NJ, USA, Tech. Rep. 2008-12, 2008.
[18] W. Backes and S. Wetzel, "Improving the parallel Schnorr–Euchner LLL algorithm," in *Proc. ICA3PP*, 2011, pp. 27–39.
[19] A. Becker, N. Gama, and A. Joux, "A sieve algorithm based on overlattices," in *Proc. ANTS*, 2014, pp. 49–70.
[20] A. Becker and T. Laarhoven, "Efficient (ideal) lattice sieving using cross-polytope LSH," Cryptol. ePrint Arch., Rep. 2015/823, 2015, pp. 1–25.
[21] J. Buchmann, E. Dahmen, and A. Hülsing, "XMSS—A practical forward secure signature scheme based on minimal security assumptions," *Post-Quantum Cryptography*, B.-Y. Yang, Ed. Berlin, Germany: Springer, 2011, pp. 117–129. [Online]. Available: https://doi.org/10.1007/978-3-642-25405-5_8, doi: 10.1007/978-3-642-25405-5_8
[22] C. Bischof, J. Buchmann, Ö. Dagdelen, R. Fitzpatrick, F. Göpfert, and A. Mariano, *Nearest Planes in Practice*. Cham, Switzerland: Springer, 2015, pp. 203–215.
[23] J. Blömer and S. Naewe, "Sampling methods for shortest vectors, closest vectors and successive minima," *Theor. Comput. Sci.*, vol. 410, no. 18, pp. 1648–1665, Apr. 2009.
[24] M. Blott, "Reconfigurable future for HPC," in *Proc. Int. Conf. High Perform. Comput. Simulation (HPCS)*, Jul. 2016, pp. 130–131.
[25] A. Blum, A. Kalai, and H. Wasserman, "Noise-tolerant learning, the parity problem, and the statistical query model," *J. ACM*, vol. 50, no. 4, pp. 506–519, Jul. 2003.
[26] P. van Emde-Boas, "Another NP-complete partition problem and the complexity of computing short vectors in a lattice," Dept. Math. Inst., Univ. Amsterdam, Amsterdam, The Netherlands, Tech. Rep. 81-04, 1981.
[27] J. W. Bos, M. Naehrig, and J. van de Pol, "Sieving for shortest vectors in ideal lattices: A practical perspective," Cryptol. ePrint Arch. 2014/880, 2014.
[28] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical hardness of learning with errors," in *Proc. 45th Annu. ACM Symp. Theory Comput. (STOC)*, 2013, pp. 575–584.
[29] J. Buchmann *et al.*, "Creating cryptographic challenges using multi-party computation: The LWE challenge," in *Proc. ASIAPKC*, 2016, pp. 11–20.
[30] J. Buchmann, R. Lindner, and M. Rückert, "Explicit hard instances of the shortest vector problem," in *Proc. 2nd Int. Workshop Post-Quantum Cryptogr. (PQCrypto)*, Cincinnati, OH, USA, Oct. 2008, 2008, pp. 79–94.
[31] J. Buchmann and C. Ludwig, "Practical lattice basis sampling reduction," *Proc. 7th Int. Symp. Algorithmic Number Theory (ANTS-VII)*, Berlin, Germany, Jul. 2006, pp. 222–237.
[32] Y. Chen and P. Q. Nguyen, *BKZ 2.0: Better Lattice Security Estimates*. Berlin, Germany: Springer, 2011, pp. 1–20.
[33] F. J. G. Correia, "Assessing the hardness of SVP algorithms in the presence of CPUs and GPUs," M.S. thesis, School Eng., Dept. Inform., Univ. Minho, Braga, Portugal, 2014.
[34] F. Correia, A. Mariano, A. Proença, C. Bischof, and E. Agrell, "Parallel improved Schnorr–Euchner enumeration SE++ on shared and distributed memory systems, with and without extreme pruning," *J. Wireless Mobile Netw., Ubiquitous Comput., Dependable Appl.*, vol. 7, no. 4, pp. 1–19, Dec. 2016.
[35] F. Correia, A. Mariano, A. Proença, C. H. Bischof, and E. Agrell, "Parallel Improved Schnorr–Euchner Enumeration SE++ for the CVP and SVP," in *Proc. 24th Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process. (PDP)*, Crete, Greece, Feb. 2016, pp. 596–603.
[36] E. Crockett and C. Peikert, "Ring-LWE challenges," Cryptol. ePrint Arch. 2014/880, 2017, pp. 1–39. [Online]. Available: https://eprint.iacr.org/
[37] Ö. Dagdelen and M. Schneider, "Parallel enumeration of shortest lattice vectors," in *Proc. 16th Int. Euro-Par Conf. Euro-Par Parallel Process.*, Ischia, Italy, Aug./Sep. 2010, pp. 211–222.
[38] J. Detrey, G. Hanrot, X. Pujol, and D. Stehlé, "Accelerating Lattice Reduction with FPGAs," in *Progress in Cryptology—LATINCRYPT* (Lecture Notes in Computer Science), vol. 6212. M. Abdalla and P. S. L. M. Barreto. Berlin, Germany: Springer, 2010, pp. 124–143. [Online]. Available: https://doi.org/10.1007/978-3-642-14712-8_8, doi: 10.1007/978-3-642-14712-8_8.
[39] L. Feng, H. Liang, S. Sinha, and W. Zhang, "HeteroSim: A heterogeneous CPU-FPGA simulator," *IEEE Comput. Archit. Lett.*, vol. 16, no. 1, pp. 38–41, Jan./Jun. 2016.
[40] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comput.*, vol. 44, no. 170, p. 463, Apr. 1985.
[41] M. Fukase and K. Kashiwabara, "An accelerated algorithm for solving SVP based on statistical analysis," *J. Inf. Process.*, vol. 23, no. 1, pp. 67–80, 2015.
[42] N. Gama, P. Q. Nguyên, and O. Regev, "Lattice enumeration using extreme pruning," in *Proc. EUROCRYPT*, 2010, pp. 257–278.
[43] N. Gama and P. Q. Nguyen, "Finding short lattice vectors within Mordell's inequality," in *Proc. STOC*, 2008, pp. 207–216.
[44] O. Garcıa-Morchón, D. Gómez-Pérez, J. Gutiérrez, R. Rietman, B. Schoenmakers, and L. Tolhuizen, "HIMMO: A lightweight collusion-resistant key predistribution scheme," Cryptol. ePrint Arch. 2014/698, 2015, pp. 1–28. [Online]. Available: https://eprint.iacr.org/
[45] A. Ghasemmehdi and E. Agrell, "Faster recursions in sphere decoding," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3530–3536, Jun. 2011.
[46] O. Goldreich, S. Goldwasser, and S. Halevi, "Public-key cryptosystems from lattice reduction problems," in *Proc. 17th Annu. Int. Cryptol. Conf. Adv. Cryptol. (CRYPTO)*, London, U.K., 1997, pp. 112–131.
[47] G. Hanrot, X. Pujol, and D. Stehlé, *Analyzing Blockwise Lattice Algorithms Using Dynamical Systems*. Berlin, Germany: Springer, 2011, pp. 447–464.
[48] T. L. Harris, "A pragmatic implementation of non-blocking linked-lists," in *Proc. DISC*, Berlin, Germany, 2001, pp. 300–314.
[49] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic Number Theory* (Lecture Notes in Computer Science), J. P. Buhler, Ed. Berlin, Germany: Springer-Verlag, 1998, pp. 267–288. pp. 267–288. [Online]. Available: https://doi.org/10.1007/BFb0054868, doi: 10.1007/BFb0054868
[50] T. Ishiguro, S. Kiyomoto, Y. Miyake, and T. Takagi, "Parallel Gauss Sieve algorithm: Solving the SVP challenge over a 128-dimensional ideal lattice," in *Proc. Int. Workshop Public-Key Cryptography (PKC)*, 2014, pp. 411–428.
[51] R. W. L. Jones, G. A. Stewart, C. Leggett, and B. M. Wynne, "Evolution of the ATLAS software framework towards concurrency," *J. Phys., Conf. Ser.*, vol. 608, no. 1, p. 012037, 2015.
[52] R. Kannan, "Improved algorithms for integer programming and related lattice problems," in *Proc. 15th Annu. ACM Symp. Theory Comput. (STOC)*, New York, NY, USA, 1983, pp. 193–206.
[53] E. Kirshanova, A. May, and F. Wiemer, "Parallel implementation of BDD enumeration for LWE," Cryptol. ePrint Arch., Tech. Rep. 2016/380, 2016.
[54] P.-C. Kuo *et al.*, "Extreme enumeration on GPU and in clouds," in *Proc. Int. Workshop Cryptogr. Hardw. Embedded Syst. (CHES)*, Nara, Japan, Sep./Oct. 2011, pp. 176–191.
[55] T. Laarhoven, "Sieving for shortest vectors in lattices using angular locality-sensitive hashing," in *Proc. CRYPTO*, 2015, pp. 3–22.
[56] T. Laarhoven, "Finding closest lattice vectors using approximate Voronoi cells," Cryptol. ePrint Arch. 2016/888, 2016. [Online]. Available: https://eprint.iacr.org/
[57] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, no. 4, pp. 515–534, 1982.
[58] R. Lindner and C. Peikert, *Better Key Sizes (and Attacks) for LWE-Based Encryption*. Berlin, Germany: Springer, 2011, pp. 319–339.

[59] M. Liu and P. Q. Nguyen, *Solving BDD by Enumeration: An Update*. Berlin, Germany: Springer, 2013, pp. 293–309.
[60] V. Lyubashevsky, "Lattice-based identification schemes secure under active attacks," in *Proc. Int. Workshop Public Key Cryptogr.*, 2008, pp. 162–179.
[61] V. Lyubashevsky, C. Peikert, and O. Regev, *On Ideal Lattices and Learning With Errors Over Rings*. Berlin, Germany: Springer, 2010, pp. 1–23.
[62] A. Mariano, T. Laarhoven, and C. Bischof, "A parallel variant of LDSieve for the SVP on lattices," in *Proc. 25th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Mar. 2017, pp. 23–30.
[63] A. Mariano, "High performance algorithms for lattice-based cryptanalysis," Ph.D. dissertation, Tech. Univ. Darmstadt, Darmstadt, Germany, 2016.
[64] A. Mariano and C. Bischof, "Enhancing the scalability and memory usage of HashSieve on multi-core CPUs," in *Proc. PDP*, Feb. 2016, pp. 545–552.
[65] A. Mariano and C. H. Bischof, "Enhancing the scalability and memory usage of HashSieve on multi-core CPUs," in *Proc. 24th Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process. (PDP)*, Heraklion, Greece, Feb. 2016, pp. 545–552. [Online]. Available: http://dx.doi.org/10.1109/PDP.2016.31
[66] A. Mariano, F. Correia, and C. Bischof, "A vectorized, cache efficient LLL implementation," in *Proc. 12th Int. Meet. High Perform. Comput. Comput. Sci.*, Porto, Portugal, Jun. 2016, pp. 162–173.
[67] A. Mariano, M. Diener, C. H. Bischof, and P. O. A. Navaux, "Analyzing and improving memory access patterns of large irregular applications on NUMA Machines," in *Proc. 24th Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process. (PDP)*, Heraklion, Greece, Feb. 2016, pp. 382–387.
[68] A. Mariano, T. Laarhoven, and C. Bischof, "Parallel (probable) lock-free hash sieve: A practical sieving algorithm for the SVP," in *Proc. 44th Int. Conf. Parallel Process. (ICPP)*, Sep. 2015, pp. 590–599.
[69] A. Mariano, S. Timnat, and C. Bischof, "Lock-free GaussSieve for linear speedups in parallel high performance SVP calculation," in *Proc. SBAC-PAD*, Oct. 2014, pp. 278–285.
[70] D. Micciancio, "Efficient reductions among lattice problems," in *Proc. 19th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA)*, Philadelphia, PA, USA, 2008, pp. 84–93. [Online]. Available: http://dl.acm.org/citation.cfm?id=1347082.1347092
[71] D. Micciancio and C. Peikert, "Hardness of SIS and LWE with small parameters," Cryptol. ePrint Arch. 2013/069, 2013. [Online]. Available: https://eprint.iacr.org/
[72] D. Micciancio and O. Regev, "Lattice-based cryptography," in *Post-Quantum Cryptography*. Berlin, Germany: Springer, 2009, pp. 147–191.
[73] D. Micciancio and P. Voulgaris, "A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations," in *Proc. STOC*, 2010, pp. 351–358.
[74] D. Micciancio and P. Voulgaris, "Faster exponential time algorithms for the shortest vector problem," in *Proc. SODA*, 2010, pp. 1468–1480.
[75] D. Micciancio and M. Walter, "Practical, predictable lattice basis reduction," Cryptol. ePrint Arch. 2015/1123, 2015. [Online]. Available: https://eprint.iacr.org/
[76] D. Micciancio and M. Walter, "Practical, predictable lattice basis reduction," in *Advances in Cryptology—EUROCRYPT*, vol. 9665. Berlin, Gernay: Springer, 2016, pp. 820–849. [Online]. Available: https://doi.org/10.1007/978-3-662-49890-3_31, doi: 10.1007/978-3-662-49890-3_31.
[77] B. Milde and M. Schneider, "A parallel implementation of GaussSieve for the shortest vector problem in lattices," in *Proc. PaCT*, 2011, pp. 452–458.
[78] I. Morel, D. Stehlé, and G. Villard, "H-LLL: Using householder inside LLL," in *Proc. ACM ISSAC*, 2009, pp. 271–278.
[79] O. Mutlu, "Memory scaling: A systems architecture perspective," in *Proc. 5th IEEE Int. Memory Workshop*, May 2013, pp. 21–25.
[80] R. Nair, "Evolution of memory architecture," *Proc. IEEE*, vol. 103, no. 8, pp. 1331–1345, Aug. 2015.
[81] P. Q. Nguyen and D. Stehlé, "An LLL algorithm with quadratic complexity," *SIAM J. Comput.*, vol. 39, no. 3, pp. 874–903, 2009.
[82] P. Q. Nguyen and J. Stern, "The two faces of lattices in cryptology," in *Proc. CaLC*, 2001, pp. 146–180.
[83] A. May, *The LLL Algorithm: Survey and Applications*, 1st ed. P. Q. Nguyen and B. Vallée, Eds. Berlin, Germany: Springer, 2009. [Online]. Available: https://doi.org/10.1007/978-3-642-02295-1_10, doi: 10.1007/978-3-642-02295-1_10.
[84] P. Q. Nguyen and T. Vidick, "Sieve algorithms for the shortest vector problem are practical," *J. Math. Cryptol.*, vol. 2, no. 2, pp. 181–207, 2008.
[85] *NVIDIA QUADRO P6000*, Nvidia, Santa Clara, CA, USA, 2016.
[86] *White Paper: NVIDIA Tesla P100*, Nvidia, Santa Clara, CA, USA, Jun. 2017. [Online]. Available: https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf
[87] A. Odlyzko, "Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature scheme," *IEEE Trans. Inf. Theory*, vol. 30, no. 4, pp. 594–601, Jul. 1984.
[88] A. M. Paulo Martins and L. Sousa, "A survey on fully homomorphic encryption: An engineering perspective," *ACM Comput. Surv.*, to be published.
[89] C. Peikert, V. Vaikuntanathan, and B. Waters, "A framework for efficient and composable oblivious transfer," in *Proc. 28th Annu. Conf. Cryptol., Adv. Cryptol. (CRYPTO)*, Berlin, Germany, 2008, pp. 554–571.
[90] K. Pietrzak, *Cryptography From Learning Parity With Noise*. Berlin, Germany: Springer, 2012, pp. 99–114.
[91] T. Plantard and M. Schneider, "Creating a challenge for ideal lattices," Cryptol. ePrint Arch. 2013/039, 2013, pp. 1–17. [Online]. Available: https://eprint.iacr.org/
[92] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *ACM SIGSAM Bull.*, vol. 15, no. 1, pp. 37–44, 1981.
[93] O. Regev, "The learning with errors problem (invited survey)," in *Proc. IEEE 25th Annu. Conf. Comput. Complex.*, Jun. 2010, pp. 191–204.
[94] O. Regev, *Lattice-Based Cryptography*. Berlin, Germany: Springer, 2006, pp. 131–141.
[95] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 34:1–34:40, 2009.
[96] R. Rietman. (2016). *Collusion Attack on HIMMO*. [Online]. Available: https://www.himmo-scheme.com/resources/collusionattack.pdf
[97] M. Schneider, "Sieving for short vectors in ideal lattices," in *Proc. AFRICACRYPT*, 2013, pp. 375–391.
[98] C.-P. Schnorr, "A hierarchy of polynomial time lattice basis reduction algorithms," *Theor. Comput. Sci.*, vol. 53, no. 2, pp. 201–224, 1987.
[99] C.-P. Schnorr, "Lattice reduction by random sampling and birthday methods," in *Proc. STACS*, 2003, pp. 145–156.
[100] C.-P. Schnorr, "Progress on LLL and lattice reduction," in *The LLL Algorithm: Information Security and Cryptography*. Berlin, Germany: Springer, 2010, pp. 145–178.
[101] C.-P. Schnorr, "Factoring integers by CVP algorithms," in *Number Theory and Cryptography* (Lecture Notes in Computer Science), vol. 8260. Berlin, Germany: Springer, 2013, pp. 73–93. [Online]. Available: https://doi.org/10.1007/978-3-642-42001-6_6, doi: 10.1007/978-3-642-42001-6_6
[102] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, no. 1, pp. 181–199, Aug. 1994.
[103] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci. (SFCS)*, Washington, DC, USA, 1994, pp. 124–134. [Online]. Available: http://dx.doi.org/10.1109/SFCS.1994.365700
[104] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997. [Online]. Available: http://dx.doi.org/10.1137/S0097539795293172
[105] D. Stehlé, "Floating-point LLL: Theoretical and practical aspects," in *The LLL Algorithm: Information Security and Cryptography*. Berlin, Germany: Springer, 2010, pp. 179–213.
[106] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa, "Efficient public key encryption based on ideal lattices," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2009, pp. 617–635.
[107] J. van de Pol, "Lattice-based cryptography," M.S. thesis, Tech. Univ. Eindhoven, Eindhoven, The Netherlands, 2011.
[108] E. Viterbo and E. Biglieri, "Computing the Voronoi cell of a lattice: The diamond-cutting algorithm," *IEEE Trans. Inf. Theory*, vol. 42, no. 1, pp. 161–171, Jan. 1996.
[109] S.-Y. Yang, P.-C. Kuo, B.-Y. Yang, and C.-M. Cheng, "Gauss sieve algorithm on GPUs," *Cryptographer's Track, RSA Conference* (Lecture Notes in Computer Science), vol. 10159, H. Handschuh, Ed. Feb. 2017, pp. 39–57.

[110] P. Q. Nguyen, "Cryptanalysis of the goldreich-goldwasser-halevi cryptosystem from crypto 1997," in *Advances in Cryptology—CRYPTO* (LNCS), vol. 1666, M. J. Weiner, Ed. Berlin, Germany: Springer, 1999, pp. 288–304.

ARTUR MARIANO received the Ph.D. degree from the Darmstadt University of Technology in 2016. He currently holds a post-doctoral position at the Instituto de Telecomunicações, University of Coimbra, and the Darmstadt University of Technology. He is involved in lattice-based cryptanalysis, with a particular focus on understanding the efficiency of attacks on modern, high-end computer architectures. He is currently involved in multi-disciplinary national and international projects aiming at understanding and improving the practicability of lattice-based cryptanalysis. In 2018, he will take on a DFG Post-Doctoral Grant researching high-performance lattice-based cryptography and is looking to establishing his own research group.

THIJS LAARHOVEN received the B.Sc. and M.Sc. degrees from the Eindhoven University of Technology, in 2009 and 2011, respectively, graduating with honors in Industrial and Applied Mathematics. He is currently a postdoctoral researcher at IBM Research, Zürich (Rüschlikon), Switzerland. Previously, he was a Ph.D. student at the Eindhoven University of Technology, studying collusion-resistant fingerprinting and group testing algorithms, lattice algorithms for solving hard lattice problems, and nearest-neighbor search techniques, such as locality-sensitive hashing (LSH) and locality-sensitive filtering (LSF).

FÁBIO CORREIA received the B.S. and M.S. degrees in informatics engineering from the University of Minho in 2012 and 2014, respectively. After completing the M.S. degree, he started working at the Institute for Scientific Computing, Darmstadt University of Technology as a Research Assistant. His work focused on developing efficient and parallel implementations of various algorithms that are used in lattice-based cryptanalysis. His research interests include high performance computing, parallel programming and lattice-based cryptanalysis. Currently, he is a Software Engineer at eXXcellent solutions, Ulm, Germany.

MANUEL RODRIGUES received the Integrated Master in Electrical and Computer Engineering degree from the University of Coimbra in 2015. Since then, he has been with the Instituto de Telecomunicações in Coimbra as a researcher. His work focuses on GPU architectures and parallel computing methods.

GABRIEL FALCÃO (S'08–M'10–SM'14) received the M.Sc. degree in electrical and computer engineering from the University of Porto and the Ph.D. degree from the University of Coimbra. In 2011 and 2012, he was a Visiting Professor with EPFL, Switzerland. He is currently an Assistant Professor with the University of Coimbra and a Researcher with the Instituto de Telecomunicações. His research interests include parallel computer architectures, GPU- and FPGA-based accelerators, digital communications, and signal processing. He is a Member of the IEEE Signal Processing Society and of the HiPEAC Network of Excellence.