International Journal on Advanced Science Engineering Information Technology

Vol.14 (2024) No. 3
ISSN: 2088-5334

A Study on the Average Number of LLL-based Using Statistical Learning

Kyunghwan Song [a], Yun Am Seo [b,*]

[a] Department of Mathematics, Jeju National University,Jeju-si, 63243, Republic of Korea
[b]Department of Data Science, Jeju National University,Jeju-si, 63243, Republic of Korea
Corresponding author: [*]seoya@jejunu.ac.kr

Abstract— **Lattice-based Cryptography is known as one of the key technologies in modern cryptography. This encryption scheme has the basis vectors from the lattice as the public key and a short-length vector in the lattice consisting of an integer combination of the basis vectors as the secret key. To break this encryption, we need to solve the Shortest Vector Problem (SVP), known as NP-hard. Therefore, instead of finding the shortest vector, LLL algorithm is often used to find a vector of sufficiently short length to break the encryption. The LLL algorithm is a well-known method for breaking this encryption, but there is still no clear answer to the question of how many times the LLL algorithm needs to be used to obtain the desired level of secret key, the average number of the (δ, η)-LLL bases in dimension n is a tool to measure the probability that the LLL algorithm solves the SVP. We can expect that this number indicates how many times the appropriate algorithm should run. There is a formula for this, but it contains some functions that take a long time to compute. We apply linear regression to the formula of the average number of the (δ, η)-LLL bases in dimension n, and therefore we obtain some formulas to approximate the average number of the (δ, η)-LLL is based on dimension n, which contains simple functions. When the dimensions are high, our model is much better regarding the computation time.**

Keywords—**Shortest vector; LLL-reduction algorithm; linear regression.**

Manuscript received 10 Sep. 2023; revised 14 Dec. 2023; accepted 24 Feb. 2024. Date of publication 30 Jun. 2024.

I. INTRODUCTION

Lattice-based cryptography represents a cryptographic framework grounded in the computational complexity of lattice-based problems, as initially expounded by Ajtai [1]. This paradigm embodies a prospective avenue within the domain of post-quantum cryptography. Notably, its utility extends to conventional cryptographic paradigms such as key exchange and digital signatures, as underscored by Nejatollahi et al. [2]. Moreover, this cryptographic approach exhibits considerable promise across diverse applications, including the realm of the Internet of Things (IoT), as articulated by Khalid et al. [3], and the sphere of medical data analytics, as elucidated by Kocabas and Soyata [4]. A foundation in linear algebra is imperative to comprehend the underpinnings of lattice-based cryptography effectively. Specifically, elucidating the concept of the span of a subset within a vector space serves as the inaugural step in this mathematical journey.

Recent research on this lattice encryption has taken a variety of forms. For example, there have been various studies on its mathematical properties and efficient algorithms [5]-[10], some variations [11], electronic voting [12], Blockchains [13], and its protection from fault or DPA [14], [15], and other invasions using the techniques of masking [16]. Furthermore, research on the hardware structure is ongoing [17], and their applications are used in the design of qTESLA [18], [19].

In the context of lattice theory, the Shortest Vector Problem (SVP) is a prominent challenge that garners attention. Identifying a vector possessing notable brevity within a lattice is significant as it emerges as a compelling contender for private key instantiation within the framework of lattice-based cryptography, as elucidated by Hoffstein et al. [20]. The algorithm known as LLL (Lenstra–Lenstra–Lovász), expounded upon by Lenstra, Lenstra, and Lovász in their seminal work [21], facilitates the efficient computation of reduced bases within a polynomial time complexity framework. This culminates in deriving a collection of vectors characterized by their brevity, rendering them apt for employment on a lattice basis. The resultant basis, termed an LLL-reduced basis, embodies a cornerstone outcome of the

LLL algorithm and has subsequently engendered several divergent iterations and adaptations.

There are so many variations of LLL algorithm. For example, the modified greedy LLL algorithm paralleled the greedy LLL algorithm [22], [23]. Furthermore, the LLL algorithm is used in Lattice-based cryptography and in many other fields. For example, L algorithm is used to construct half-Hadamard matrices [24], solve the hidden subset sum problem [25], compute multidimensional theta functions [26], construct Hermite Normal Form [27], detect periodicity in digital images [28], attack ECDSA [29].

In this paper, we use the statistical techniques of linear regression are employed in the investigation of the average number of (δ, η)-LLL bases within the n-dimensional space. The outcomes of this investigation yield a series of analytical expressions, encapsulating elementary functional relationships, that serve as approximations for the aforementioned average counts of (δ, η)-LLL bases in n-dimensional contexts. Particularly noteworthy is the marked efficiency exhibited by our proposed model, particularly in scenarios characterized by high dimensions, as it significantly outperforms conventional methodologies in terms of computational time requirements.

We start with a brief introduction of Lattice based cryptography.

Let us consider the simplest two-dimensional case. [30]

- To receive the message, Alice creates a public key, which is a sufficiently large natural number q.
- And then Alice creates two secret keys f and g that are satisfying $f < \sqrt{q/2}, \sqrt{q/4} < g < \sqrt{q/2}$, and f and qg are relatively prime.
- After that Alice creates another public key $h \equiv f^{-1}g (\mathrm{mod}\ q)$ and then publish the pair of public key (q, h).
- Bob wants to send a message $m < \sqrt{q/4}$ to Alice without being seen by Eve.
- So, Bob compute $e \equiv rh + m\ (\mathrm{mod}\ q), 0 < e < q$ with a random number r and send the ciphertext e to Alice.
- Alice computes $a \equiv fe\ (\mathrm{mod}\ q)$, $0 < a < q$ and then compute $b \equiv f^{-1}a\ (mod\ g), 0 < b < g$.
- Then $b = m$.

Let us see an example with small numbers.

- Let $q = 23, f = 2,$ and $g = 3$.
- Then $h \equiv 2^{-1} \cdot 3 \equiv 12 \cdot 3 \equiv 13\ (mod\ 23)$.
- Let plaintext $m = 2$. With a random number $r = 6$, Bob computes $e \equiv 6 \cdot 13 + 2 \equiv 11\ (\mathrm{mod}\ 23)$ and send e.
- Alice computes $a \equiv 2 \cdot 11 \equiv 22\ (\mathrm{mod}\ 23)$, and compute $b \equiv 2^{-1} \cdot 22 \equiv 2 \cdot 1 \equiv 2\ (mod\ 3)$ and $b = m = 2$.

Eve can try a brute-force attack, but it is not useful. So, Eve tries to find the private key (f, g) from the public key (q, h). If Eve knows a pair (F, G) such that$Fh \equiv G\ (\mathrm{mod}\ q), F = O(\sqrt{q}), and\ G = O(\sqrt{q})$, then (F, G) behaves as if it were secret key. It is clear that $Fh \equiv G\ (\mathrm{mod}\ q)$ and $Fh = G + qR$ with an unknown integer R are equivalent. Then finding (F, G) is equivalent to solve $F(1, h) - R(0, q) = (F, G)$ with known vectors $(1, h)$, $(0, q)$ and unknown parameters $F, G,$ and R. So, solving Lattice-based cryptography is equivalent to this problem: Find a sufficiently nonzero small vector between the integer combinations of known vectors. As the dimension increases, this problem becomes much more difficult because the number of integer combinations to consider increases dramatically. This feature has been used to ensure the security of Lattice-based cryptography in sufficiently high dimensions.

LLL algorithm, as introduced in Section 1, is an efficient method to find a sufficiently small vector by reducing given basis within a polynomial time complexity. From Eve's point of view, it is important to know how many times this algorithm can be used to get a sufficiently small nonzero vector, and from Alice's or Bob's point of view, they need an indicator that they should finish sending messages before running the LLL algorithm that sufficient times.

Unfortunately, there is not yet enough research on this algorithm to know how many tries it takes to get a reasonably good small nonzero vector. We are simply utilizing the LLL algorithm in a way that allows us to run it a few times and see if we get a good vector. However, as the dimension is higher, you're going to have to run these algorithms a really large number of times, and it's going to take longer and longer to verify that the result of the algorithm is a reasonably short vector. So, there's definitely a need for research on how many runs are the answer for the range of good choices.

In this paper, we use the following notation to define the set of integer combinations of basis vectors.

$$span_{\mathbb{Z}}(S) = \left\{ \sum_{i=1}^{n} a_i v_i : a_1, \ldots, a_n \in \mathbb{Z} \right\} \quad (1)$$

where $S = \{v_1, v_2, \ldots, v_n\}$.

Let $\mu_{i,j} = < v_i, \frac{v_j^*}{||v_j^*||^2} >$, $v_1^* = v_1$ and $v_i^* = v_i - \sum_{j=1}^{i-1} \mu_{i,j} v_j^*$ for $i \geq 2$.

Note that $\{v_1^*, v_2^*, \ldots, v_n^*\}$ is an orthogonal basis and if this process is possible for given vector space, then we can find the shortest vector because at least one of v_i^* can be shorten to a shortest vector by just multiplying a scalar. But in general, $\mu_{i,j} = < v_i, \frac{v_j^*}{||v_j^*||^2} >$ is not an integer for a lattice base $\{v_1, v_2, \ldots, v_n\}$. Therefore, in the original LLL algorithm and its variants, use round (μ) instead of μ.

For the purpose to see the quality of the output of LLL algorithm, we first check the pseudocode of the LLL algorithm as follows: [30]

- Input: a basis $\{v_1, v_2, \ldots, v_n\}$for $\mathbb{R}^n$, $\frac{1}{2} \leq \eta \leq \delta \leq 1$,
  Output: a (δ, η)-LLL basis $\{v_1, v_2, \ldots, v_n\}$.
- $k = 2$
- while $k \leq n$,
  - for $i = k - 1$ to 1,
    - $v_k = v_k - round(\mu_{k,i}) v_i$
  - $If\ \delta||v_{k-1}^*|| \leq ||v_k^* + \mu_{k,k-1} v_{k-1}^*||$
    - $k = k + 1$
  - else
    - $\mathrm{tmp} = v_{k-1}$
    - $v_{k-1} = v_k$
    - $v_k = \mathrm{tmp}$
    - $k = min(k - 1, 2)$
- Output $\{v_1, \ldots, v_n\}$, a (δ, η)-LLL basis.

The output, a (δ,η)-LLL basis satisfies the following conditions: [31]

- $|\mu_{i,j}| \leq \eta$ for all $j < i$.
- $\delta \|v_i^*\| \leq \|v_{i+1}^* + \mu_{i+1,i} v_i^*\|$ for all $i = 1,\ldots,n-1$

Note that the output can vary for the same lattice by starting with different basis vectors. Also, for any fixed lattice, a basis that contains all the shortest vectors is not unique in general. For example, let $v_i = e_i$. Then $\{v_1, v_2, \ldots, v_n\}$ is a basis for $\mathbb{Z}^n$ and the number of bases for $\mathbb{Z}^n$ that contains all the shortest vector is 2^n because $v_i = \pm e_i$ for any i, generates $\mathbb{Z}^n$. Therefore, if the possible number of outputs for fixed LLL is sufficiently close to 2^n, then by using LLL algorithm, user expects that the output basis contains the shortest vector(s). For this reason, there are some results for the number of LLL basis and the author of [30] obtain the average number of the (δ,η)-LLL bases in dimension n. That is,

$$2(2\eta)^{(n-1)(n-2)/2} \prod_{i=2}^{n} \frac{S_i(1)}{\zeta(i)} \frac{1}{n} \prod_{i=1}^{n-1} \frac{1}{i(n-i)} \times \prod_{i=1}^{n-1} \int_{-\eta}^{\eta} \sqrt{\delta^2 - x^2}\, dx \tag{2}$$

where $S_i(x)$ is the surface area of a sphere in $\mathbb{R}^i$ of radius x. The formula is the exact value of the average number but computing this value takes so long time as n increases so it is beautiful but somewhat impractical. This paper aims to find an approximated formula for equation (2) that contains just simple functions, such as exponential functions.

First, we start by simplifying the given equation (2) into equation (3). In equation (3), $\xi(i) = \frac{1}{2} i(i-1)\pi^{-i/2}\Gamma(\frac{i}{2})\zeta(i)$, *the Riemann-Xi function*, a variation of the Riemann-zeta function. So that one of the product forms was deleted. After that, we compute the above formula to obtain that the exponent part of this formula is about cubic. We check this fact by estimation with a regression model.

$$= 2(2\eta)^{(n-1)(n-2)/2} \prod_{i=2}^{n} \frac{\frac{2\pi^{\frac{i}{2}}}{\Gamma\left(\frac{i}{2}\right)}}{\zeta(i)} \frac{1}{n} \prod_{i=1}^{n-1} \frac{1}{i(n-i)} \times \prod_{i=1}^{n-1} \int_{-\eta}^{\eta} \sqrt{\delta^2 - x^2}\, dx$$

$$= 2(2\eta)^{(n-1)(n-2)/2} \prod_{i=2}^{n} \frac{1}{\frac{1}{2} i(i-1)\pi^{-\frac{i}{2}}\Gamma\left(\frac{i}{2}\right)\zeta(i)} \times \prod_{i=1}^{n-1} \int_{-\eta}^{\eta} \sqrt{\delta^2 - x^2}\, dx \tag{3}$$

which is equal to

$$2^{n^2-3n+\frac{4}{2}} \eta^{\frac{(n-1)(n-2)}{2}} \prod_{i=2}^{n} \frac{1}{\xi(i)} \times \prod_{i=1}^{n-1} \int_{-\eta}^{\eta} \sqrt{\delta^2 - x^2}\, dx$$

II. Materials and Method

A. Material

The following experimental combinations of n, δ, and η in formula (2) were used to produce input and output data for the LLL algorithm. There were 18 experimental combinations of n in increments of 10 from 30 to 200, 5 experimental combinations of δ in increments of 0.5 from 0.75 to 0.95, and 3 experimental combinations of η: 0.51, 0.52, and 0.53. The total number of combinations in the experiment is 18×5×3=270 combinations.

If we set η to 0.53, δ to 0.95, and n to 200, the calculated value of equation (1) is about 2.81e+118532, which exceeds the number of floating-point digits of the computer, and the result becomes infinite. Therefore, in this study, we used the "Rmpfr" and "gmp" libraries of the R program, which can calculate large integers. These libraries require a large computational cost to produce results.

Fig. 1 shows the time taken to compute equation (2) as a function of δ and n after fixing η to 0.53. For all δ, we see that there is a difference in the computation time when the dimensionality is small, but as the dimensionality increases, there is no significant difference in the computation time. The computation time scales linearly with the number of dimensions.

The purpose of this study is to learn a kind of meta-model for equation (2) by statistical methods or machine learning to produce a value that approximates the true value faster than the number of possible bases resulting from the calculation of the LLL algorithm, and to produce similar results. However, although it is possible to produce input and output data for learning, it is not possible to learn a model in the usual way due to the difficulty in processing the value of the dependent variable, which is the prediction target, due to the floating-point problem of the computer.

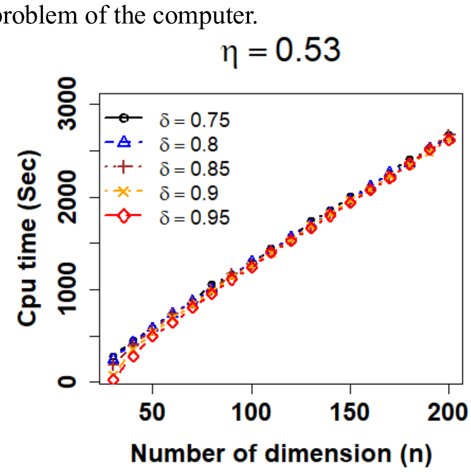

Fig. 1 LLL basis computation time for δ and dimension number combinations.

B. Method

To solve the problem mentioned in the previous section, this study proposes a method that separates the integer and exponential parts of a value that exceeds the number of floating-point digits, predicts them separately, and combines the two prediction results. The exponential part is known to be proportional to n^3, so we use it to learn with a linear regression model, while the integer part is filled with as many numbers as the computer's floating-point digits, so it is difficult to find any rules for the dimensionality. To solve this problem, we propose a method that cuts out numbers below a certain number of digits from the integer data to be trained and adds digits proportional to the number of dimensions to predict approximate values, but not true values. Also, since it predicts approximate values for true values, it predicts an interval with a (1-α)% confidence interval for the predicted value.

To summarize the above algorithm

- Step1: Separate dependent variable values into integer and exponential parts
- Step2: Predict the exponent part by computing $y = f(x) + \varepsilon$.
- Step3: Scale the number of digits in the integer part of the data to be proportional to the dimension
- Step4: By using $\log(y) = f(x) + \varepsilon$ predict the integer part and inverse transform
- Step5: Combine the predicted integer part and exponent part

We fix the values of δ and η in the algorithm suggested in this study. $\eta = 0.53$, and $\delta = 0.95$.

III. Results and Discussion

The relationship between the exponential part and the integer part is in the form of a cubic polynomial as shown in Fig. 2. Therefore, the exponential part was predicted by fitting the polynomial regression model in equation (4). The regression coefficients in the regression model were estimated using the least squares method.

$$y = \beta_0 + \beta_1 x^3 + \beta_2 x^2 + \beta_3 x^{\square} + \varepsilon \qquad (4)$$

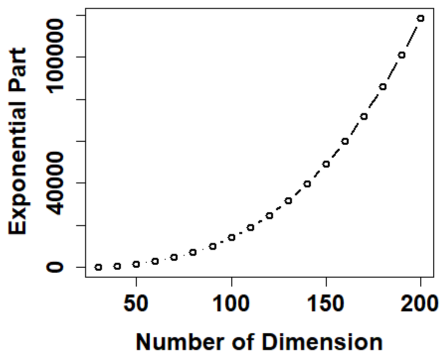

Fig. 2 Relationship between exponential parts and number of dimensions.

After fitting the model, the corrected coefficient of determination was 1, and the regression coefficients for each term were all statistically significant at the 0.001 level of significance (α). Fig. 3 is a scatter plot of the true and predicted values. The exponential part of the prediction was almost identical to the true value.

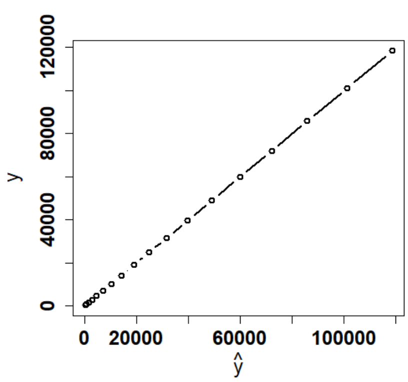

Fig. 3 Scatterplot of predicted exponential part and true values

Table shows the 95% confidence intervals for the predicted values of the exponential regression model. In all dimensions, we predicted intervals containing the true value, with a range of 18 on average, which is very close to the true value, and the time required for this prediction is less than one second.

TABLE I
95% Confidence Interval for the Predicted Exponential Part

Number of dimension (n)	Lower bound	Exact value	Upper bound
30	306	323	327
40	801	809	820
50	1632	1636	1650
60	2892	2897	2910
70	4677	4683	4695
80	7079	7088	7097
90	10195	10205	10213
100	14117	14129	14135
110	18940	18953	18958
120	24758	24771	24776
130	31666	31677	31684
140	39757	39767	39775
150	49127	49134	49145
160	59868	59874	59886
170	72076	72081	72094
180	85844	85850	85862
190	101267	101275	101285
200	118437	118453	118458

Fig. 4 is a scatter plot of the predicted and true values of the integer part. The modified coefficient of determination is 0.998, which is slightly lower than the prediction performance of the exponential part, but it is analyzed that it is possible to predict the interval of the true value with some accuracy. Table 2 shows the 95% confidence intervals for the integer partial predictions. We see that the confidence intervals for all dimensions contain true value. Combining the integer interval estimates with the exponential interval estimates, the output of the LLL algorithm was able to approximate the interval with the true value for the number of possible bases.

TABLE II
95% Confidence Interval for Predicted Integer Part

Number of dimension (n)	Lower bound	Exact value	Upper bound
30	6.63e+77	2.43e+78	1.53e+79
40	7.34e+77	3.89e+78	1.42e+79
50	7.87e+77	7.44e+78	1.37e+79
60	8.24e+77	1.83e+78	1.35e+79
70	8.47e+77	2.66e+78	1.36e+79
80	8.62e+77	3.06e+78	1.39e+79
90	8.73e+77	8.38e+78	1.42e+79
100	8.84e+77	3.17e+78	1.45e+79
110	8.98e+77	1.61e+78	1.49e+79
120	9.16e+77	1.63e+78	1.52e+79
130	9.37e+77	7.01e+78	1.54e+79
140	9.62e+77	3.73e+78	1.56e+79
150	9.87e+77	9.22e+78	1.59e+79
160	1.01e+78	5.03e+78	1.62e+79
170	1.02e+78	3.51e+78	1.68e+79
180	1.01e+78	2.16e+78	1.76e+79
190	9.81e+77	9.53e+78	1.90e+79
200	9.22e+77	2.81e+78	2.12e+79

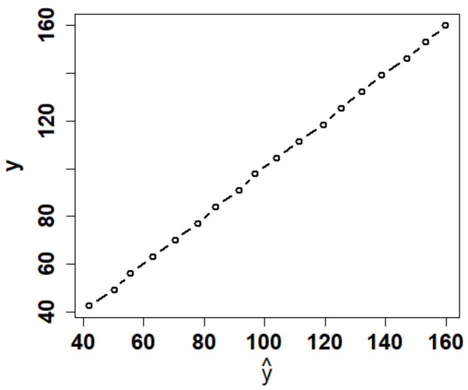

Fig. 4 Scatterplot of predicted integer part and true values.

IV. Conclusion

For more accurate prediction in the future, we believe that the performance will be improved by applying a learning method that can accurately predict the integer part. Also, we try to expand the result to arbitrary $\boldsymbol{\delta}$ and $\boldsymbol{\eta}$, not fixed. As you can see in fig. 1, the value of the formula (1) is dominated by $\mathbf{n}$, not $\boldsymbol{\delta}$ or $\boldsymbol{\eta}$. So, the method used in this paper will have the effect to predict for arbitrary $\boldsymbol{\delta}$ and $\boldsymbol{\eta}$. From these predictions, we suggest an indicator that shows how many times LLL algorithm has to be executed to find sufficiently short vector of given fixed lattice. We intend to follow up on this work in the future so that we can come up with appropriate metrics for similarly conceptually well-defined things that are still difficult to use in practice. We believe that many areas have not been solved mathematically or experimentally, especially when limited to areas that require a lot of computation, such as high dimensions, and we would like to continue research on this area because practicality also depends on this area.

Similarly, there are many things that are not yet well understood mathematically in lattice-based cryptosystems. For example, for the BKZ algorithm, which performs basis reduction similarly to the LLL algorithm, the appropriate number of runs to obtain a good basis is not well defined. Mathematical and statistical characterization of these practical issues for different algorithms is another topic of future research related to this study.

Acknowledgment

This research was supported by the 2023 scientific promotion program funded by Jeju National University.

References

[1] M. Ajtai, "Generating hard instances of lattice problems", Proc. Electron. Colloq. Comput. Complexity, 1996.

[2] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, R. Cammarota, “Software and hardware implementation of lattice-cased cryptography schemes”, Center for Embedded Cyber-Physical Systems, pp. 1-43, 2017.

[3] A. Khalid, S. McCarthy, M. O’Neill, and W. Liu, “Lattice-based Cryptography for IoT in A Quantum World: Are We Ready?,” 2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI), Jun. 2019, doi: 10.1109/iwasi.2019.8791343.

[4] Ö Kocabaş and T Soyata, “Medical data analytics in the cloud using homomorphic encryption”, In E-Health and Telemedicine: Concepts, Methodologies, Tools, and Applications. IGI Global, 2016.

[5] M. N. John, O. G. Udoaka, I. U. Udoakpan “Group Theory in Lattice-Based Cryptography.”, Int. J. Math. And Appl., vol. 11, no. 4, pp. 111–125, Dec. 2023.

[6] M. F. Esgin, R. Steinfeld, J. K. Liu, and D. Liu, “Lattice-Based Zero-Knowledge Proofs: New Techniques for Shorter and Faster Constructions and Applications,” Lecture Notes in Computer Science, pp. 115–146, 2019, doi: 10.1007/978-3-030-26948-7_5.

[7] C. Baum and A. Nof, “Concretely-Efficient Zero-Knowledge Arguments for Arithmetic Circuits and Their Application to Lattice-Based Cryptography,” Public-Key Cryptography – PKC 2020, pp. 495–526, 2020, doi: 10.1007/978-3-030-45374-9_17.

[8] V. Lyubashevsky, N. K. Nguyen, and M. Plançon, “Lattice-Based Zero-Knowledge Proofs and Applications: Shorter, Simpler, and More General,” Lecture Notes in Computer Science, pp. 71–101, 2022, doi:10.1007/978-3-031-15979-4_3.

[9] L. Ducas and W. van Woerden, “On the Lattice Isomorphism Problem, Quadratic Forms, Remarkable Lattices, and Cryptography,” Lecture Notes in Computer Science, pp. 643–673, 2022, doi: 10.1007/978-3-031-07082-2_23.

[10] S. Lyu, L. Liu, C. Ling, J. Lai, and H. Chen, “Lattice codes for lattice-based PKE,” Designs, Codes and Cryptography, vol. 92, no. 4, pp. 917–939, Nov. 2023, doi: 10.1007/s10623-023-01321-6.

[11] N. Alkeilani Alkadri, R. El Bansarkhani, and J. Buchmann, “BLAZE: Practical Lattice-Based Blind Signatures for Privacy-Preserving Applications,” Lecture Notes in Computer Science, pp. 484–502, 2020, doi: 10.1007/978-3-030-51280-4_26.

[12] D. F. Aranha, C. Baum, K. Gjøsteen, T. Silde, and T. Tunge, “Lattice-Based Proof of Shuffle and Applications to Electronic Voting,” Lecture Notes in Computer Science, pp. 227–251, 2021, doi:10.1007/978-3-030-75539-3_10.

[13] C. Ma and M. Jiang, “Practical Lattice-Based Multisignature Schemes for Blockchains,” IEEE Access, vol. 7, pp. 179765–179778, 2019, doi:10.1109/access.2019.2958816.

[14] D. Heinz and T. Pöppelmann, “Combined Fault and DPA Protection for Lattice-Based Cryptography,” IEEE Transactions on Computers, vol. 72, no. 4, pp. 1055–1066, Apr. 2023, doi:10.1109/tc.2022.3197073.

[15] T. Oder, “Efficient and side-channel resistant implementation of lattice-based cryptography”, Doctoral dissertation, Dissertation, Bochum, Ruhr-Universität Bochum, 2019.

[16] J.-P. D’Anvers, D. Heinz, P. Pessl, M. Van Beirendonck, and I. Verbauwhede, “Higher-Order Masked Ciphertext Comparison for Lattice-Based Cryptography,” IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 115–139, Feb. 2022, doi:10.46586/tches.v2022.i2.115-139.

[17] W. Tan, A. Wang, Y. Lao, X. Zhang, K. K. Parhi, “Low-latency VLSI architectures for modular polynomial multiplication via fast filtering and applications to lattice-based cryptography”, arXiv preprint arXiv:2110.12127, 2021.

[18] W. Wang, S. Tian, B. Jungk, N. Bindel, P. Longa, and J. Szefer, “Parameterized Hardware Accelerators for Lattice-Based Cryptography and Their Application to the HW/SW Co-Design of qTESLA,” IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 269–306, Jun. 2020, doi:10.46586/tches.v2020.i3.269-306.

[19] E. Alkim, P. S. L. M. Barreto, N. Bindel, J. Krämer, P. Longa, and J. E. Ricardini, “The Lattice-Based Digital Signature Scheme qTESLA,” Lecture Notes in Computer Science, pp. 441–460, 2020, doi:10.1007/978-3-030-57808-4_22.

[20] J. Hoffstein, J. Pipher, J.H. Silverman, “An introduction to mathematical cryptography”, vol. 1, New York: Springer, 2008, doi:10.1007/978-0-387-77993-5.

[21] A. K. Lenstra, H. W. Lenstra, and L. Lovász, “Factoring polynomials with rational coefficients,” Mathematische Annalen, vol. 261, no. 4, pp. 515–534, Dec. 1982, doi: 10.1007/bf01457454.

[22] L. Chen, Z. Xing, Y. Li, and S. Qiu, “Efficient MIMO Preprocessor With Sorting-Relaxed QR Decomposition and Modified Greedy LLL Algorithm,” IEEE Access, vol. 8, pp. 54085–54099, 2020, doi:10.1109/access.2020.2980922.

[23] L. Chen, Y. Wang, Z. Xing, S. Qiu, Q. Wang, and Y. Zhang, “A Paralleled Greedy LLL Algorithm for 16×16 MIMO Detection,” 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Oct. 2020, doi: 10.1109/iscas45731.2020.9181195.

[24] A. Goldberger and Y. Strassler, “A practical algorithm for completing half-Hadamard matrices using LLL,” Journal of Algebraic Combinatorics, vol. 55, no. 1, pp. 217–244, Nov. 2021, doi:10.1007/s10801-021-01077-z.

[25] J.-S. Coron and A. Gini, “A Polynomial-Time Algorithm for Solving the Hidden Subset Sum Problem,” Lecture Notes in Computer Science, pp. 3–31, 2020, doi: 10.1007/978-3-030-56880-1_1.

[26] J. Frauendiener, C. Jaber, and C. Klein, “Efficient computation of multidimensional theta functions,” Journal of Geometry and Physics, vol. 141, pp. 147–158, Jul. 2019, doi:10.1016/j.geomphys.2019.03.011.

[27] G. H. Cho, H.-S. Lee, S. Lim, and Y. Kim, “Storage efficient algorithm for Hermite Normal Form using LLL,” Linear Algebra and its Applications, vol. 613, pp. 183–200, Mar. 2021, doi:10.1016/j.laa.2020.12.022.

[28] L. Hajdu, B. Harangi, A. Tiba, and A. Hajdu, “Detecting Periodicity in Digital Images by the LLL Algorithm,” Mathematics in Industry, pp. 613–619, 2019, doi: 10.1007/978-3-030-27550-1_78.

[29] K. Ryan, "Return of the hidden number problem.: A widespread and novel key extraction attack on ecdsa and dsa", vol. 2019, pp. 146-168, Nov. 2018, [online] Available: https://tches.iacr.org/index.php/TCHES/article/view/7337.

[30] Kim S. “On the shape of a high-dimensional random lattice.” Ph.D. thesis, Stanford University, 2015.