Efficient Gaussian sampling for RLWE-based cryptography through a fast Fourier transform*

Márcio Barbado Júnior[1]

[1]Departamento de Engenharia de Computação e Sistemas Digitais
Universidade de São Paulo (USP) – São Paulo, SP – Brazil

mbarbado@usp.br

Abstract. *Quantum computing threatens classical cryptography, leading to the search for stronger alternatives. The cryptographic approach based on lattices is considered as a viable option. Schemes with that approach use Gaussian sampling, a design which brings along two concerns: efficiency and information leakage. This work addresses those concerns in the RLWE formulation, for digital signatures. Efficiency mitigation uses the central limit theorem, and the Walsh–Hadamard transform, whereas the information leakage risk is reduced via isochronous implementation. Up to 2^{23} samples are queried, and the results are compared against those of a cumulative distribution table sampler. Statistical metrics show the suitability of the presented sampler in a number of contexts.*

1. Introduction

Current lattice-based cryptography derives from [Ajtai 1996], which is a work based on a hard computational problem known as short integer solution (SIS). From that point on, other hardness assumptions started being considered. One of them is the ring learning with errors problem (RLWE), which consists in an algebraically structured variant of the learning with errors problem (LWE). Basically, LWE consists in distinguishing between two types of equations: truly random ones, and those whose underlying structure have been masked by a controlled amount of noise [Regev 2009]. RLWE has an analogous formulation, the main difference being a hidden underlying structure of polynomial rings [Lyubashevsky et al. 2013]. Often, the generation of random values in RLWE-based cryptosystems make use of discrete Gaussian sampling (DGS) routines, which may suffer from efficiency issues [Dwarakanath and Galbraith 2014, Ortiz et al. 2015], and may be vulnerable to timing side-channel attacks, leading to information leakage [Bernstein et al. 2008, Alkim et al. 2019, Pöppelman et al. 2019]. Evidences of the need for efficiency and security advancements in the DGS primitive have been provided by many recent works, mostly stimulated by the post-quantum cryptography standardization process being promoted by the National Institute of Standards and Technology (NIST) [CSRC 2022].

In this work, a competitive DGS strategy is presented for RLWE-based cryptography. Regarding efficiency, the strategy relies on using the central limit theorem and a fast Fourier transform (FFT) in convenient ways. Information leakage risks are mitigated through isochronous implementation. Results are then compared against a widely-accepted DGS strategy, namely the cumulative distribution table (CDT).

*This work was supported by research grants 169284/2018-2 from Conselho Nacional de Desenvolvimento Científico e Tecnológico, BP 1600.03.18 and BP 1975.07.20 from Fundação para o Desenvolvimento Tecnológico da Engenharia, and 3372 and 300804 from Fundação de Apoio à Universidade de São Paulo.

2. Related works

Lattice-based cryptography works to address DGS may cover the sampler in a number of forms. Some works may present samplers as standalone proposals, while other works may adopt wider scopes, of which samplers take parts. Also, there are works based on previously existing DGS strategies. Authors may regard specification and implementation, the latter presenting hardware and software proposals In [Ducas et al. 2013], a digital signature scheme with its own sampler is presented. It is called the Bimodal Lattice Signature Scheme (BLISS). In [Ortiz et al. 2015], authors illustrate the inherent side-channel problem of DGS by implementing and testing previously existing strategies in software forms.

Earlier works with embedded and original DGS strategies are [Alkim et al. 2019], presenting the qTESLA digital signature scheme, [Alkim et al. 2020], presenting the FrodoKEM key encapsulation scheme, and [Howe et al. 2019], presenting the FALCON digital signature scheme. All of these have joined the NIST competition [CSRC 2022].

For reference, a comparison of sampling strategies among some works is present in Table 1. The hereby proposed work is referred to as DiGS in the table, which is short for "discrete Gaussian sampling".

Table 1. DGS strategies for DiGS and related works.

Sampler	**Assumption**	**DGS strategy**
DiGS (ours)	RLWE	CLT and FWHT
BLISS [Ducas et al. 2013]	SIS	Inversion and rejection
FALCON [Howe et al. 2019]	NTRU	CDT and rejection
FrodoKEM [Alkim et al. 2020]	LWE	CDT
qTESLA [Alkim et al. 2019]	RLWE	CDT and rejection

Italicized text indicates participation in [CSRC 2022].

Additionally, comprehensive web pages listing DGS works in lattice-based cryptography can be found on [Micciancio 2022, Bernstein et al. 2022]. As for DGS isochronous implementation, [Reparaz et al. 2016] presents an assessment tool, available in the public domain. In [Micciancio and Walter 2018], there is a similar concern as well. Implementation security is covered for a DGS strategy based on precomputed values.

Related content about the central limit theorem (CLT) general formulation, as used herein, is present in [Dwarakanath and Galbraith 2014]. Even though the authors do not reach acceptable results, their proposal of using the referred theorem rises as an interesting contribution.

Partially related works to cover the FWHT in shuffling-related strategies are [Rader 1969, Harwit and Sloane 1979]. Both deal with random variable generation by means of the Hadamard matrix, the latter featuring the referred topic in its appendix, with a didactic approach, offering detailed examples. Harwit and Sloane also cover the relationships between Hadamard matrices and Walsh functions, the Hadamard transform, and its fast variant.

3. Concepts

This section presents an overview of RLWE-based cryptography, focusing on its DGS primitive and a few algorithms to build it. It also presents the CLT and FWHE, which form the basis for the sampling strategy adopted by this work. Finally, timing side-channel vulnerabilities, the security concern of this work, are briefly depicted.

3.1. RLWE-based cryptography

In lattice-based cryptography, an RLWE scheme structures the LWE problem in polynomial rings. Those schemes hold high asymptotic efficiency, and strong provable security, backed by the computational complexity of lattice problems [Peikert 2014].

3.1.1. Polynomial rings

A polynomial ring is an algebraic structure consisting in a set of finite or infinite polynomials. The properties to be satisfied by polynomial rings are identity for addition and multiplication, associative and commutative multiplication, and distributive multiplication.

This work is interested in finite rings of s polynomials, provided with integer coefficients. Also, the s parameter implies $\mathbb{Z}[x]$ polynomials should have maximum degrees of up to $s-1$. In Equation 1, a finite polynomial ring R_q with two moduli is presented. The first modulus, x^s+1, is generic. The second modulus an RLWE polynomial ring should obey is q, which is a large public prime. $\mathbb{Z}[x]$ expresses the integer nature of coefficients.

$$R_q = \frac{\mathbb{Z}_q[x]}{x^s+1}. \tag{1}$$

3.1.2. The RLWE problem

The RLWE problem is built upon LWE, which in turn is built upon nondeterministic polynomial (NP) problems from the lattice theory, e.g., the shortest vector problem (SVP) and the closest vector problem (CVP) [Ajtai 1996, Regev 2009]. In order to illustrate the RLWE problem, an R_q polynomial ring consisting of s polynomials is considered. The polynomials are firstly made equal to zero, and then they are separated from their constant terms, allowing a matrix equation representation as that of Equation 2.

$$\begin{bmatrix} a_{1\,1} & a_{1\,2} & \dots & a_{1\,s-2} & a_{1\,s-1} & a_{1\,s} \\ a_{2\,1} & a_{2\,2} & \dots & a_{2\,s-2} & a_{2\,s-1} & a_{2\,s} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{s-2\,1} & a_{s-2\,2} & \dots & a_{s-2\,s-2} & a_{s-2\,s-1} & a_{s-2\,s} \\ a_{s-1\,1} & a_{s-1\,2} & \dots & a_{s-1\,s-2} & a_{s-1\,s-1} & a_{s-1\,s} \\ a_{s\,1} & a_{s\,2} & \dots & a_{s\,s-2} & a_{s\,s-1} & a_{s\,s} \end{bmatrix} \times \begin{bmatrix} x^{s-1} \\ x^{s-2} \\ \vdots \\ x^2 \\ x \\ 1 \end{bmatrix} = \begin{bmatrix} -a_{1\,s} \\ -a_{2\,s} \\ \vdots \\ -a_{s-2\,s} \\ -a_{s-1\,s} \\ -a_{s\,s} \end{bmatrix}. \tag{2}$$

By representing the constant terms vector as b, the solution is represented as the X vector. As LWE problems deal with artificially perturbed equations, a pseudo-random error e is added to X, as shown in Equation 3.

$$X = A^{-1} \times b \rightarrow X + e \neq A^{-1} \times b,\ e \neq 0. \tag{3}$$

Distinguishing e influence by taking equations in pairs is the RLWE problem.

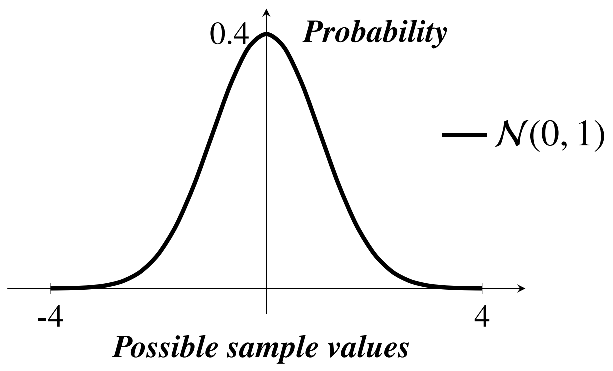
3.2. Gaussian sampling

The term Gaussian in Gaussian sampling refers to sampling from normal distributions of probabilities, which may be used for random number generation in lattice-based cryptography. A distribution of probability can be described in continuous and discrete forms. The latter can be described by the probability mass function (PMF), which is the probability density funtion (PDF) for the discrete case, and by the cumulative density function (CDF). Regarding normal distributions, there is also a compact option to represent them through their two main parameters, as in $\mathcal{N}(\mu,\ \sigma^2)$. That notation brings the μ mean and the σ^2 variance. It constitutes a simplified normal PDF, which by its turn can be described as Equation 4, according to [Weisstein 2021].

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \mathrm{e}^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2},\ x \in \mathbb{R} \tag{4}$$

A graphical example is present in Figure 1, which portrays a standard normal distribution PDF, bounded inside the $[-4,\ 4]$ sampling domain. Being standard implies mean 0 and variance 1.

Figure 1. A [-4, 4] bounded-domain view of a standard normal distribution PDF.



3.3. Sampling strategies

Current sampling strategies pertinent to DGS in lattice-based cryptography often adopt inversion, rejection and CDT, as seen in Table 1. Inversion is a transform algorithm which turns a probability value p into a number whose probability in another distribution is at most p, while rejection follows the premise that computations not achieving certain criteria should be discarded. The CDT algorithm, used by this work as reference, samples from a precomputed table, the cumulative distribution table. It does so by taking a random real value $r \in [0, 1)$, and performing a search operation on the table, comparing r to each value in there. The last value to be bigger than r has its index s returned as a valid observation for the sample. Formally, the algorithm tries to find an s index such that the $\text{CDT}_{s-1} < r < \text{CDT}_s$ condition is satisfied. Its main drawback being the high memory consumption involved, as the algorithm requires a precomputed table. An isochronous CDT construction, which fills a table and samples from it, can be found in [Alkim et al. 2019].

3.4. Central limit theorem

The central limit theorem (CLT) refers to a number of enunciations. The core idea behind most of them is that by sampling randomly and repeatedly from a supposedly uniform population distribution, and then computing an average for each sample, the resulting distribution of averages tends to a normal distribution. Samples correspond to uncorrelated uniform random variables, and the obtained distribution of averages is a normal random variable [Rader 1969]. Thus, the CLT can also be used to estimate values, e.g., as a maximum likelihood estimator (MLE). In this work, the CLT is used to accelerate the generation of Gaussian random variables, e.g., seemingly normal distributions, in a more efficient way. The $\mu_{\mathfrak{D}}$ mean of 2^β random variables approaches the theoretical population average as the number of samples grows.

A supposedly uniform population distribution is to be sampled, and in that lies a part of the CLT acceleration used, precisely, in population choice. The chosen population is previously manipulated, forcing the original, supposedly uniform population distribution to be normal already. Considering the samples obtained in this work, it is desirable to understand how far results are from a trustable normal distribution.

3.5. The fast Walsh–Hadamard transform

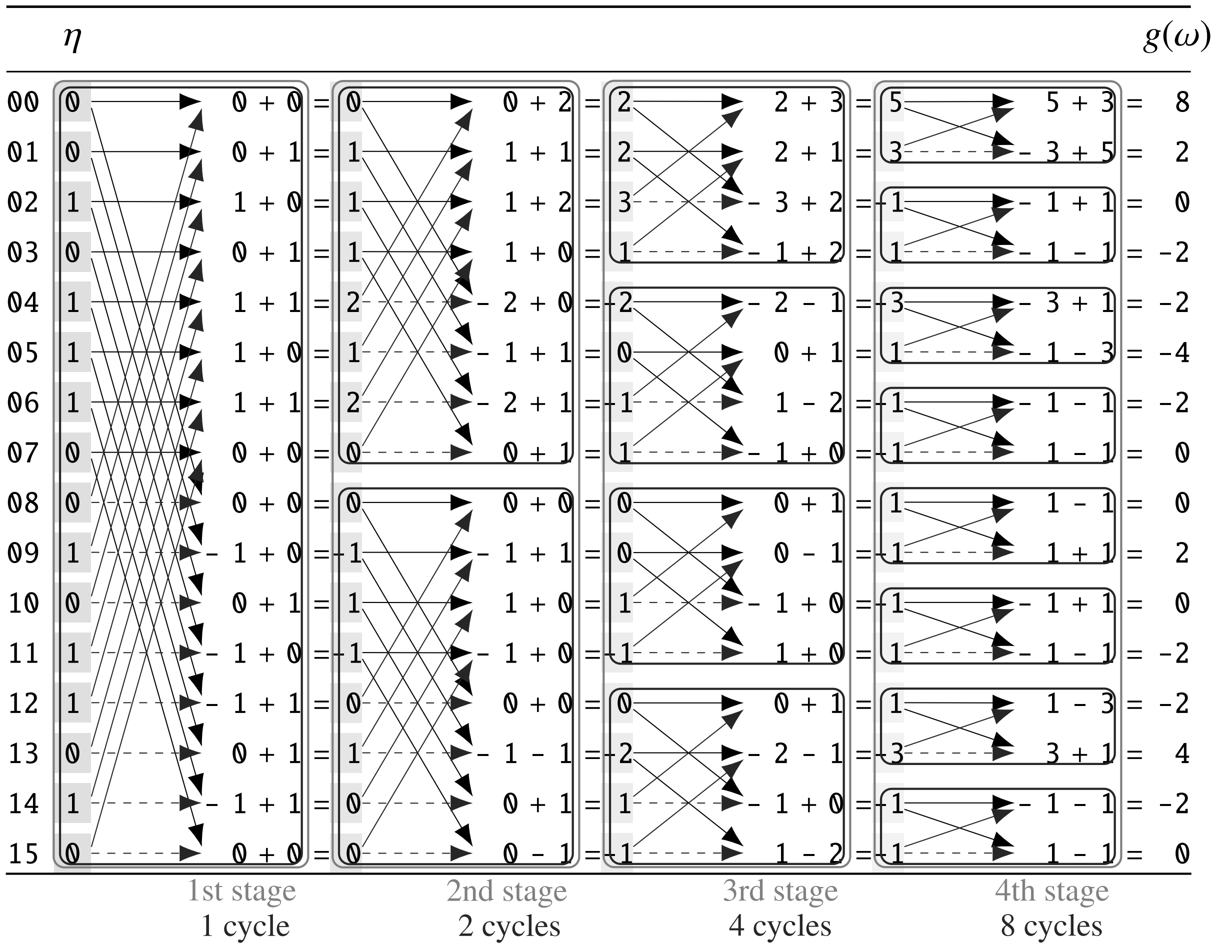
The fast Walsh–Hadamard transform is an $O(n \log n)$ FFT algorithm, basically constituting an operation of multiplication between an Hadamard matrix and an input matrix η. In this work, the latter is an initialization vector (IV), e.g., $\eta = (0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0)$. An Hadamard matrix H is a symmetric matrix whose elements are either one or minus one [Sylvester 1867, Rader 1969]. An example of order four is presented in Equation 5.

$$H_{\beta=2} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{5}$$

The number of steps needed to compute an FWHT of order n is approximately $n \log_2 n$ [Harwit and Sloane 1979], or simply $n\beta$, considering the main input variable of this work.

The diagram in Figure 2 illustrates the transform evolution for the previously mentioned η vector. The resulting FWHT is presented in column $g(\omega)$. Continuous arrows generate positive numbers, and dashed arrows generate negative numbers. A stage can be understood as a vector shuffling procedure. There are $\beta = 4$ stages, and in each stage, 2^4 operations are computed, one for each vector position.

Figure 2. Diagram of FWHT with stages and cycles marked.



3.6. Timing side-channel vulnerabilities

Timing side-channel vulnerabilities were firstly described in [Kocher 1996]. The reffered work shows that, once many cryptographic routine implementations do not run in fixed times, due to performance and other reasons, there is a number of possible attacks which can be carried out against such implementations. Those attacks can exploit slight differences in computing times of the vulnerable cryptosystems. The referred differences can be measured and analysed, allowing attackers to obtain secret keys.

Mitigation measures can be implemented in application, operating system, and hardware levels. That can be accomplished through isochronous architectures for example. Isochrony in coding can be described as a design pattern which imposes an invariant execution time to a given routine. Specifically regarding DGS isochronous construction in RLWE-based cryptography, an example can be seen in the qTESLA scheme [Alkim et al. 2019].

4. FWHT Gaussian sampler

DiGS, the discrete Gaussian sampler presented herein is an LWE-oriented DGS construction. It addresses efficiency through CLT convergence acceleration and an FFT algorithm, namely, the FWHT. It also mitigates timing-attack risks through an isochronous design for the referred algorithm.

Building of approximate Gaussians is achieved by means of sampling algebraic Gaussian distributions, instead of uniform distributions, meaning a biased procedure is enforced. This is also the phase of CLT convergence, which consists in getting to a probability distribution fitting in reduced time. The number of sampling iterations are reduced by a precision acceptability criteria, which estimates tolerance values for differences between the mean of the polynomial roots and the roots of the algebraic polynomial, offering opportunities for routines to be terminated prematurely, still being effective. The FWHT is then used for shuffling. Sampling is always performed in a fixed regime, i.e., Gaussian parameter values do not change during a sampling session.

4.1. The variables

Variables may have an $\mathcal{N}$ index if they relate to the algebraic normal distribution PDF, or a $\mathcal{D}$ index if they relate to the approximated Gaussian PMF. Additionally, the following presentation of variables divides them in two groups, input and output, all of them are presented in the following sections. Values of input variables are supposed to be secret.

Input variable: ${\sigma_{\mathcal{N}}}^2$ denotes the variance for a Gaussian PDF. Its square root is the standard deviation σ, which is used to determine where to bound the PDF, for practical purposes.

Input variable: $\tau_{\mathcal{N}}$ is the tail-cut factor. It is not often used for simple Gaussian descriptions, but it constitutes an artificial resource for bound manipulation.

Input variable: β is a scalar, $\beta \in \mathbb{N}^*$, created to comply with dimension requirements of Hadamard matrices. This is because the FWHT-based sampling strategy demands a 2^β number of positions on its input vector. The variable influences all of the sampling routines, both on PDF and PMF functions. As a consequence, not only does the β exponent dictate the number of stages in the FWHT algorithm presented by this work, but it also determines a 2^β order square matrix, the full-rank lattice dimension to work with, and the number of keys to be supported. In practical terms, β influences the sample size, that is, the number of observations in each sample, and it also influences the number of samples itself. Thus, a β value of 8 produces a sample size and a number of samples of 256. Regarding the sample size, denoted in this work as s, each observation corresponds to one polynomial coefficient, that is, a sample size of 256 implies a polynomial of degree 255 with 256 coefficients. As for the number of samples, each sample represents a polynomial, so 256 samples stand for 256 polynomials.

Output variable: $\sigma_{\mathcal{D}}$ denotes the standard deviation for the Gaussian PMF.

Output variable: $\mu_{\mathfrak{D}}$ denotes the mean of a normal distribution PMF.

Output variable: s lowercase s is the even 2^{β} sample size to be used both by PDF and PMF sampling routines, and it also represents a security parameter of the RLWE problem. The sample size is the number of observations in each sample, and, in this work, that also means the number of coefficients in each RLWE polynomial. Bigger values are expected to reduce tails, and make statistical distance smaller between algebraic PDF and approximated PMF.

Output variable: $skewness_{\mathfrak{D}}$ denotes quality. An approximated normal distribution PMF whose skewness value is closer to zero indicate a more likely bell-shaped function.

Output variable: $kurtosis_{\mathfrak{D}}$ as $skewness_{\mathfrak{D}}$, the $kurtosis_{\mathfrak{D}}$ variable stands as a measure of quality. Higher values of kurtosis indicate lower-quality PMF approximation.

Output variable: $CPUcycles_{\mathcal{N}}$ portrays an efficiency metric, reporting the number of CPU cycles for a complete sampling routine.

Output variable: $CPUcycles_{\mathfrak{D}}$ portrays an efficiency metric, reporting the number of CPU cycles spent in the production of a noise vector.

Output variable: $CPUcycles_{total}$ portrays an efficiency metric, corresponding to the total CPU cycles, from the moment routines start trying to achieve CLT convergence until the production of a noise vector.

Output variable: $entropy$ is also known as the Shannon entropy. For the purposes of this work, higher values of this variable are desirable, as they offer more randomness.

Output variable: D_{KL} is the relative entropy, measured in bits. Values closer to zero are desirable in this work, as they stand for PMF functions better representing the original PDF function.

4.2. The algorithm

The FWHT routine presented is supposed to be provided with argumentative values for the following parameters: the exponent β, the $\sigma_{\mathcal{N}}$ standard deviation for the Gaussian PDF, the $\tau_{\mathcal{N}}$ tail-cut factor, and the $option$ sampling algorithm. As for the algorithm variables, ℓ is expected to store the number of cycles on a given stage. Variable ℓl is expected to store the current cycle length, that is, the number of vector positions the current cycle takes. Variable ℓlh is expected to store half the cycle length, and it also helps didactically in the pseudocode, emphasizing the idea cycles always have two equal-sized chunks, one for addition operations and other for subtraction operations. Variable v is a 2^{β}-sized vector, which firstly stores an η initialization vector, produced by the

$sample()$ function, and the PDF sampling strategy to be performed by this function is arbitrated by the $option$ parameter. Variable v_{aux} serves the purpose of preserving the value stored in vector position $\ell_{aux}\ell l + \ell l_{aux}$ before it is altered. Variables β_{aux}, ℓ_{aux}, and ℓl_{aux} help in counting and controlling the number of iterations in each of the three loop structures. Variable ℓp is expected to store a relative position in vector v, always in the first chunk of the current cycle, computed as $\ell_{aux}\ell l + \ell l_{aux}$.

Algorithm 1 An FWHT isochronous construction.

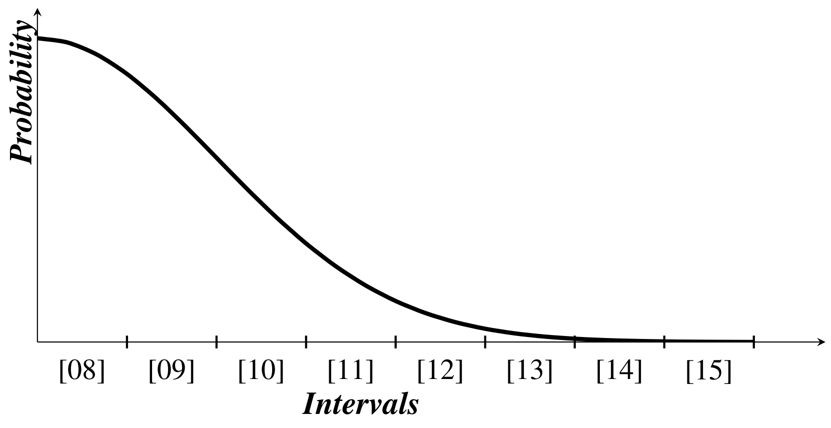
```
algorithm fwht_non_recursive ($\beta$: natural, $\sigma_{\mathcal{N}}$: real, $\tau_{\mathcal{N}}$: real, $option$: natural)
    $\ell$:                    natural; // Number of cycles on a given stage.
    $\ell l$:                   natural; // Current cycle length.
    $\ell lh$:                  natural; // Half the cycle length.
    $\ell p$:                   natural; // Position in the first chunk of the current cycle.
    $v[2^\beta]$, $v_{aux}$:          integer;
    $\beta_{aux}$, $\ell_{aux}$, $\ell l_{aux}$:   natural;

    $v$                     $\leftarrow sample(\beta, \sigma_{\mathcal{N}}, \tau_{\mathcal{N}}, option)$;
    $\ell$                     $\leftarrow 1$;
    $\ell l$                    $\leftarrow 2^\beta$;

    for ($\beta_{aux}$ from $0$ to $\beta - 1$, step 1)
        $\ell$                 $\leftarrow \ell \cdot 2^{\beta_{aux}}$;
        $\ell l$                $\leftarrow \ell l / 2^{\beta_{aux}}$;
        $\ell lh$               $\leftarrow \ell l / 2$;
        for ($\ell_{aux}$ from $0$ to $\ell$, step 1)
            for ($\ell l_{aux}$ from $0$ to $\ell lh - 1$, step 1)
                $\ell p$          $\leftarrow \ell_{aux}\ell l + \ell l_{aux}$;
                $v_{aux}$         $\leftarrow v[\ell p]$;
                $v[\ell p]$       $\leftarrow v_{aux} + v[\ell p + \ell lh]$;
                $v[\ell p + \ell lh]$ $\leftarrow v_{aux} - v[\ell p + \ell lh]$;
            end for
        end for
    end for

    return $v$;
end algorithm
```

The first loop structure iterates β times, corresponding to the number of stages. The second loop structure iterates on the number of cycles of the current stage. In the first stage, there is only one cycle. Then, in the second stage, there are two cycles, and that value keeps being doubled at each new stage. The third and last loop is based on the length of the current cycle, but it iterates upon only half that value. In the first stage, the cycle length equals 2^{β}. In the second stage, the cycle length becomes $2^{\beta-1}$. Then $2^{\beta-2}$ in the third stage, and the exponent keeps being decremented by one at each new stage, until it reaches the value of one, meaning the last stage has cycles with 2 elements. As the number of cycles double, the number of elements in each cycle decreases in an exponential rate. Using only half the cycle length for iteration is due to the FWHT always breaking a cycle length into two equal-sized chunks. In this work, a chunk corresponds to half the length of a cycle. For each chunk element, there is a sign attribution, followed by an addition or a subtraction operation, respectively represented by lines 21 and 22 of Algorithm 1. Line 21 represents the first of the two chunks, so its first position corresponds to the first position of the current cycle, which is determined by ℓp. In the first chunk, vector elements are given plus signs, and then each of those elements is used in two operations, the first operation occurring in the first chunk itself, and the second operation occurring in the second chunk. Line 22 represents the second of the two chunks, so its first position corresponds to the middle of the current cycle, which is determined by $\ell p + \ell lh$. In the second chunk, vector elements are given plus signs for operations in the first chunk, and minus signs for operations in the second chunk itself.

Lastly, v is returned as the $g(\omega)$ transform output. It has the same 2^{β} length of the η input, and its content is a set of integer coefficients for a polynomial. Those integer values may be turned into binaries if necessary.

In the diagram of Figure 2 didacticism is addressed, and many one-valued positions are used in the η vector. However, for the sampling routine of this work, there is a single non-zero value in the referred IV, all other positions being occupied by zeros. The η initialization vector is obtained from a preliminary Gaussian sampling observation. Thus, still considering the $\beta = 4$ example, the referred sampling is performed in the positive portion of a $\mathcal{N}(0, \sigma^2)$ truncated PDF, whose domain is divided into 16 equally-sized intervals.

Figure 3. The intervals to be observed for $\beta = 4$.



Each interval of Figure 2 corresponds to a position in η. The μ mean assumes a value of zero, and the positive bound is $\tau\sigma$. Observed values assume alternating signs minus and plus. In practice, half the number of intervals is used, as in Figure 3, and the alternating signs simulate a $(-\tau\sigma,\ \tau\sigma]$ domain. Intervals corresponding to positions [00], [01], [02], [03], [04], [05], [06], and [07] are not shown because they stand in the negative side of the curve. A negative [08] corresponds to [07], a negative [09] corresponds to [06], a negative [10] corresponds to [05], a negative [11] corresponds to [04], a negative [12] corresponds to [03], a negative [13] corresponds to [02], a negative [14] corresponds to [01], and a negative [15] corresponds to [00]. After having one of its positions occupied by 1, η is subjected to the FWHT, in order to generate a transform representation.

5. Tests and results

The tests are executed with the CDT and FWHT strategies, consisting in a Python routine, developed from a SageMath script written by Jefferson E. Ricardini. The continuous-case algebraic normal probability distribution used is $\mathcal{N}(0, 14.71025358)$. Table 2 presents the results for the CDT sampling strategy . For all of the listed β values, skewness is zero and kurtosis is -2. Table 3 presents the results for the FWHT sampling strategy . For all of the listed β values, the obtained μ, is zero. The same applies to skewness values. A histogram plot corresponding to FWHT tests is present in Figure 4. All of the relative entropy values are truncated for cleaner presentation.

Table 2. Results for the CDT sampling strategy.

β	s	**CPU cycles** $\times 10^6$	$\mu_{\mathfrak{D}}$	$\sigma_{\mathfrak{D}}$	**relative entropy**
11	2048	13.49	-13.0	8.0	0.360
12	4096	13.41	-4.5	10.5	0.353
13	8192	13.60	-1.0	10.0	0.366
14	16384	13.31	-22.0	1.0	0.360
15	32768	13.33	4.0	26.0	0.355
16	65536	14.35	-2.0	2.0	0.359
17	131072	14.13	-3.5	23.5	0.363
18	262144	14.34	-12.5	7.5	0.359
19	524288	14.20	3.5	2.5	0.359
20	1048576	14.19	-17.0	15.0	0.360

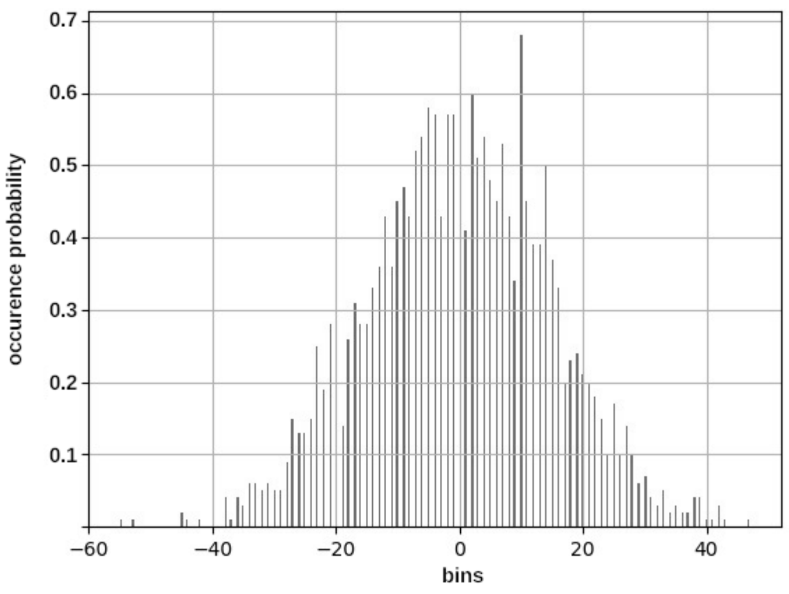
6. Conclusion and final remarks

As for the input parameters the β exponent determines the most relevant aspects of the sampler, from lattice dimension, and polynomial degrees, to CLT convergence rate, and the entropy of samples. Additionally, in terms of implementation, β is directly responsible for the CPU cycles spent. Modeling DGS via the β parameter alone reveals itself as a challenge in searching for optimal values, because higher values raise entropy, but they also lower efficiency. Thus, β represents a trade-off involving efficiency and security. The standard deviation $\sigma_{\mathcal{N}}$ influences inflection points of the function and the cardinality of the sample space. For the settings of this work, the optimal standard deviation value of the PDF is $\sigma_{\mathcal{N}} = 14.71025358$. The tail-cut factor $\tau_{\mathcal{N}}$ of the PDF was initially planned to be used as an input parameter because, along with $\sigma_{\mathcal{N}}$, it influences the cardinality of sample spaces. However, acceptable efficiency results are obtained without it, thus, for simplicity, this work chooses not to use it.

Table 3. Results for the FWHT sampling strategy.

β	s	CPU cycles $\times 10^3$	$\sigma_{\mathcal{D}}$	kurtosis	relative entropy
11	2048	96.18	2	-2	0.333
12	4096	104.16	14	-2	0.332
13	8192	110.67	12	-2	0.325
14	16384	124.11	10	-2	0.316
15	32768	114.24	10	-2	0.320
16	65536	111.72	6	-2	0.319
17	131072	111.51	15	-2	0.320
18	262144	109.83	8	-2	0.317
19	524288	111.72	0	-3	0.318
20	1048576	109.20	18	-2	0.317

Figure 4. Histogram of observations for the FWHT sampling strategy.



As for the outputs feasible lattice dimensions of up to 2^{23}, are tested by this work Regarding $\sigma_{\mathcal{D}}$ values, in most cases, the obtained numbers are more than 10% above or below $\sigma_{\mathcal{N}}$. That is not considered as a reasonably good result for this work. Regarding the Kullback–Leibler divergence values, the fact that all measures are closer to zero than they are to one denotes a reasonable approximation achievement. Such values indicate PMFs resemble their respective PDFs. As for the quality of the PMFs, the obtained skewness values are zero, showing maximum quality is achieved for that metric, and Kurtosis is always -2 in all tests but one. Approaching zero from the left, with absolute values below 3 means the PMFs are platykurtic, i.e., they are flatter than they should be.

As for the efficiency comparison between FWHT and CDT firstly, it is relevant to mention the algorithm complexities involved. While CDT is $O(\tau_{\mathcal{N}} \cdot \sigma_{\mathcal{N}})$, the FWHT shuffling routine is $O(n \cdot \log n)$, or $O(\beta \cdot 2^{\beta})$ in terms of the β parameter. CDT sampling complexity, being governed by τ, the tail-cut factor, and σ, the standard deviation, depends on the size of the sampling interval. FWHT sampling complexity is governed by the β exponent, responsible for the number of samples, the lattice dimension, and other variables. In the scope of this work, a more efficient algorithm is the one which takes less central processing unit (CPU) cycles to securely complete the noise generation routine for a given parameter setup. Considering the values of β in the interval $4 \leq \beta \leq 20$, and the further settings of this work, the results of efficiency for the FWHT strategy are better than the CDT strategy results. The difference, measured in CPU cycles, tends to be two orders of magnitude, in favor of the FWHT approach. As for the values of $\mu_{\mathfrak{D}}$, which are supposed to follow the $\mu_{\mathcal{N}}$ value of zero, that is observed in all of the FWHT tests but it is not the case for the CDT tests. This issue, regarding the mean of the Gaussian, is consistent with the relative entropy results, which are generally bigger for CDT tests. Thus, the relative entropy values show that the PMF vector is closer to the algebraic function as the FWHT strategy is used. In the tests performed, the only case of similar Kullback–Leibler values for CDT and FWHT is $\beta = 7$. The values of $\mu_{\mathfrak{D}}$ following those of $\mu_{\mathfrak{D}}$ denote successful acceleration of the CLT.

References

Ajtai, M. (1996). Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pages 99–108, New York, NY, USA. ACM. STOC '96.

Alkim, E., Barreto, P. S. L. M., Bindel, N., Longa, P., and Ricardini, J. E. (2019). The Lattice-Based Digital Signature Scheme qTESLA. Cryptology ePrint Archive, Report 2019/085.

Alkim, E., Bos, J. W., Ducas, L., Longa, P., Mironov, I., Naehrig, M., Nikolaenko, V., Peikert, C., and Raghunathan, A. (2020). FrodoKEM Learning With Errors Key Encapsulation Algorithm Specifications And Supporting Documentation.

Bernstein, D. J., Hallgren, S., Vollmer, U., Buchmann, J., Dahmen, E., Szydlo, M., Overbeck, R., Sendrier, N., Micciancio, D., Regev, O., Ding, J., and Yang, B.-Y. (2008). *Post Quantum Cryptography*. Springer Publishing Company, Incorporated, Berlin Heidelberg, 1st edition.

Bernstein, D. J., Lange, T., Cayrel, P.-L., and Peters, C. (2022). Lattice-based public-key cryptography. http://pqcrypto.org/lattice.html, accessed on MAY 26th, 2022.

CSRC, N. (2022). Post-quantum cryptography. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/, accessed on MAY 26th, 2022.

Ducas, L., Durmus, A., Lepoint, T., and Lyubashevsky, V. (2013). Lattice Signatures and Bimodal Gaussians. Cryptology ePrint Archive, Report 2013/383.

Dwarakanath, N. C. and Galbraith, S. D. (2014). Sampling from discrete gaussians for lattice-based cryptography on a constrained device. *Applicable Algebra in Engineering, Communications and Computing*, 25(3):159–180.

Harwit, M. and Sloane, N. J. A. (1979). Appendix hadamard and s-matrices, walsh functions, pseudo-random sequences, and the fast hadamard transform. In *Hadamard Transform Optics*, pages 200–228. Elsevier Inc.

Howe, J., Prest, T., Ricosset, T., and Rossi, M. (2019). Isochronous gaussian sampling: From inception to implementation with applications to the falcon signature scheme. pages 1–23.

Kocher, P. C. (1996). Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Koblitz, N., editor, *Advances in Cryptology - CRYPTO '96*, pages 104–113, Berlin Heidelberg. , Springer-Verlag. LNCS 1109.

Lyubashevsky, V., Peikert, C., and Regev, O. (2013). On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35.

Micciancio, D. (2022). Sampling, Lattice Cryptography. http://cseweb.ucsd.edu/ daniele/LatticeLinks/Sampling.html, accessed on MAY 26th, 2022.

Micciancio, D. and Walter, M. (2018). Gaussian sampling over the integers: Efficient, generic, constant-time. pages 01–28.

Ortiz, J. N., Aranha, D. F., and Dahab, R. (2015). Implementação em Tempo Constante de Amostragem de Gaussianas Discretas. SBSeg 2015.

Peikert, C. (2014). Lattice Cryptography for the Internet. In Mosca, M., editor, *Post-Quantum Cryptography 6th International Workshop*, Lecture Notes in Computer Science, pages 197–219, Switzerland. Springer International Publishing.

Pöppelman, T., Alkim, E., Avanzi, R., Bos, J., Ducas, L., de la Piedra, A., Schwabe, P., and Stebila, D. (2019). *NewHope Algorithm Specifications and Supporting Documentation*. c/o Thomas Pöppelmann, Infineon Technologies AG, Am Campeon 1-12, 85579 Neubiberg, Germany, version 1.03 edition. NIST.

Rader, C. M. (1969). A new method of generating gaussian random variables by computer. Technical Report 1969-49, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, Lexington, Massachusetts.

Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40.

Reparaz, O., Balasch, J., and Verbauwhede, I. (2016). Dude, is my code constant time? Cryptology ePrint Archive, Report 2016/1123.

Sylvester, J. J. (1867). Lx. thoughts on inverse orthogonal matrices, simultaneous sign-successions, and tessellated pavements in two or more colours, with applications to newton's rule, ornamental tile-work, and the theory of numbers. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 34(232):461–475.

Weisstein, E. W. (2021). Probability Density Function. From MathWorld–A Wolfram Web Resource.