Time-Independent Discrete Gaussian Sampling for Post-Quantum Cryptography


Published in:
Proceedings of the 2016 International Conference on Field-Programmable Technology (FPT '16)

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
Link to publication record in Queen's University Belfast Research Portal

Publisher rights
© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

General rights
Copyright for the publications made accessible via the Queen's University Belfast Research Portal is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The Research Portal is Queen's institutional repository that provides access to Queen's research output. Every effort has been made to ensure that content in the Research Portal does not infringe any person's rights, or applicable UK laws. If you discover content in the Research Portal that you believe breaches copyright or violates any law, please contact openaccess@qub.ac.uk.
Time-Independent Discrete Gaussian Sampling
For Post-Quantum Cryptography

A. Khalid, J. Howe, C. Rafferty, and M. O’Neill
Centre for Secure Information Technologies (CSIT), Queen’s University Belfast, UK.

Abstract—As the development of a viable quantum computer nears, existing widely used public-key cryptosystems, such as RSA, will no longer be secure. Thus, significant effort is being invested into post-quantum cryptography (PQC). Lattice-based cryptography (LBC) is one such promising area of PQC, which offers versatile, efficient, and high performance security services. However, the vulnerabilities of these implementations against side-channel attacks (SCA) remain significantly understudied. Most, if not all, lattice-based cryptosystems require noise samples generated from a discrete Gaussian distribution, and a successful timing analysis attack can render the whole cryptographic system broken, making the discrete Gaussian sampler the most vulnerable module to SCA. This research proposes countermeasures against timing information leakage with FPGA-based designs of the CDT-based discrete Gaussian samplers with constant response time, targeting encryption and signature scheme parameters. The proposed designs are compared against the state-of-the-art and are shown to significantly outperform existing implementations. For encryption, the proposed sampler is 9x faster in comparison with appropriate practical parameters for both encryption and signatures, are presented.

II. BACKGROUND
A. Lattice-based Cryptography

One foundational work that underpins LBC is the learning with errors (LWE) problem [9]. Cryptosystems based upon the LWE problem enjoy worst-case to average-case hardness; they are proven infeasible to break unless all instances of certain lattice problems are easy to solve [10], [11]. Lattice-based cryptosystems, based on the hardness of the LWE problem are as hard to solve as Definition 1. Definition 1: [LWE] For positive integers n and q ≥ 2, the secret s ∈ Z^n, and a probability distribution χ on Z_q, let A_s,χ be the LWE distribution, obtained by choosing a ← Z^n_q, a noise term e ← χ, and outputting (a, b = (a, s)+e) ∈ Z^n_q×Z_q. The decisional LWE problem is defined as, given access to m independent samples chosen according to A_s,χ, distinguish between these m LWE samples and ones chosen uniformly at random with noticeable probability.

B. Discrete Gaussian Sampling

The error distribution χ seen in Definition 1 is almost always defined as the discrete Gaussian distribution1. The centered discrete Gaussian distribution D_{0,σ}, over Z, with standard deviation σ, is defined proportionally such that a value x ∈ Z is sampled from D_σ with the probability ρ_σ(x)/ρ_σ(Z), where:

\[ \rho_σ(x) = \exp\left(\frac{-x^2}{2σ^2}\right) \quad \text{and} \quad \rho_σ(Z) = \sum_{k=-∞}^{∞} ρ_σ(k). \]

With the exception of key-exchange protocols such as [12], which are able to employ the slightly less “normal”, binomial distribution.

1
Considering $S_x = \rho_x(Z) = \sum_{k=-\infty}^{\infty} \rho_x(k) \approx \sqrt{2\pi}\sigma$, the probability of sampling $x \in \mathbb{Z}$ from the distribution $D_{x,\sigma}$ is calculated as $\rho_x(x)/S_x$.

1) Exploiting Symmetry: To half the initial memory requirement, one can consider the distribution over $\mathbb{Z}^\ast$, proportional to $\rho_x(x)$ for $\forall x > 0$. For $x = 0$, $\rho_x(0)$ is halved, otherwise this will be counted twice. The distribution can be recovered by adding a random sign bit after sampling.

2) Practical Discrete Gaussian Parameters: The statistical distance between the ”perfect” theoretical discrete Gaussian distribution and the ”practical” should be no greater than $2^{-3}$.

It is recommended [13] that precision need be no greater than $\lambda/2$, for a target security level $\lambda$-bits, as it is argued that no algorithm can distinguish between a ”perfect” sampler and one with statistical distance $2^{-\lambda/2}$. Two important cryptographic applications for the samplers are targeted: encryption and digital signature schemes. The parameters are from the encryption scheme by Lindner and Peikert [14] (LP)\(^\dagger\), where $(\sigma, \lambda, \tau) = (3.33, 64, 9.42)$, and the digital signature scheme by Ducas et al. [15] (BLISS), where $(\sigma, \lambda, \tau) = (215, 64, 9.42)$.

3) Gaussian Convolution: The standard deviation can be significantly decreased by using Peikert’s convolution lemma [16], adapted by Pöppelmann et al. [19]. Referring to [8], [19] for the formal definitions of the smoothing parameter $\eta$ and Kullback-Leibler divergence respectively, the adaption states:

**Lemma 1:** Let $x_1 \leftarrow D_{x,\sigma_1}, x_2 \leftarrow D_{x,\sigma_2}$ for some positive real $\sigma_1, \sigma_2$ and let $\sigma_3 = \sigma_1^2 + \sigma_2^2$ and $\sigma_3' = \sigma_1' + \sigma_2'$.

For any $\epsilon \in (0, 1/2)$ if $\sigma_1 \geq \eta_1(\mathbb{Z})/\sqrt{2\pi}$ and $\sigma_3 \geq \eta_3(\mathbb{Z})/\sqrt{2\pi}$, then ”perfect” distribution $\mathcal{P}$ of $x_1 + x_2$ verifies

$$D_{KL}(\mathcal{P} \| D_{x,\sigma}) \leq 2 \left( 1 - \frac{1 + \epsilon^2}{1 - \epsilon^2} \right)^2 \approx 32\epsilon^2.$$  

**Proof:** The proof of this lemma is referred to in [19].

**Lemma 1** holds for $\sigma_{\text{BLISS}} = 215$ by setting $k = 11$, such that $\sigma' = \sigma/\sqrt{1 + k^2} \approx 19.47$, and by sampling twice $x'_1, x'_2 \leftarrow D_{x,\sigma'}$ a value $x \leftarrow D_{x,\sigma}$ can be built as $x = x'_1 + kx'_2$. Although an additional sample is required, a smaller standard deviation means that memory consumption of the precomputed tables is significantly reduced; memory consumption is reduced from 130kb for $\sigma_{\text{BLISS}} = 215$, to 11.74kb for $\sigma_{\text{BLISS}} = 19.47$.

III. THE CUMULATIVE DISTRIBUTION TABLE SAMPLER

The cumulative distribution table (CDT) sampler requires a precomputed table of discrete Gaussian cumulative distribution function (CDF) values [8]. CDT sampling is arguably more promising than other discrete Gaussian sampling schemes, as the distribution parameters are known in advance. The CDF values range from $0 \leq x \leq 1$, and are stored in a look-up table $S[x]$, where the total number of table entries is at least: $N = \tau \times \sigma$, where $0 = S[0] < S[1] < \ldots < S[N-1] = 1$.

CDT sampling works as follows: sample $r \leftarrow \{ -\tau\sigma, \ldots, \tau\sigma \}$, with $\lambda$ bits of precision. $r$ is compared against the CDF table contents to find an interval such that: $S[x] \leq r < S[x+1]$.

where $x$ is output as the required discrete Gaussian sample, occurring with probability $\rho_x(x) = S[x+1] - S[x]$.

For comparisons, binary search is chosen, and is detailed in Algorithm 1. Pointers $\text{min}$, $\text{mid}$, and $\text{max}$ point to the first, last, and middle of the search space, respectively. $r$ is iteratively compared to the middle value of the search space $S[\text{cur}]$, whose upper or lower half is discarded depending on the comparison result. For a finite search space with $N$ samples, the comparisons required before a match does not exceed $\lceil \log_2(N) \rceil$.

**Algorithm 1** CDT Sampling from $D_{x,\sigma}$ via Binary Search

**Require:**
1: Three integers $\text{min}$, $\text{mid}$, and $\text{max}$.
3: Uniformly sampled $r \in \{0, \ldots, (2^b - 1)\}$ and bit $b \in \{0, 1\}$.
**Ensure:**
$\text{min} \leftarrow 0; \text{max} \leftarrow N; \text{mid} \leftarrow (\text{min} + \text{max})/2$.
4: while $(\text{max} > \text{min})$ do
5: if $(r \geq S[\text{mid}])$ then
6: $\text{min} \leftarrow (\text{mid} + 1)$;
7: else
8: $\text{max} \leftarrow \text{mid}$;
9: return $x = (-1)^b(\text{mid} - 1)$

A. Previous work

The use of CDT sampling was proposed by Peikert [8], and adapted by Ducas et al. [15]. Pöppelmann et al. [19] implemented BLISS on reconfigurable hardware and suggested optimisations for the CDT sampler, including hashing to reduce the search space and skipping the leading zero storage to reduce the table size by a factor of 2. Du and Bai [20] further optimised hardware area by using piecewise comparison and hashing. These optimisations reduce the precomputed table size, and improve throughput. However, hashing divides the search intervals into irregular sizes, meaning the binary search has non-constant execution time, making it susceptible to timing analysis attacks.

The only CDT sampler design on a FPGA with constant-time throughput is by Pöppelmann and Güneysu [21]. This fully pipelined design offers a single cycle per sample throughput, but the large number of parallel comparisons renders it impractical. Roy et al. [7] presented a hardware design of a discrete Gaussian sampler resistant against timing attacks, using a Knuth-Yao sampler that generates a batch of samples, subsequently shuffled to dissociate the related timing information. However, this design is non-constant time and only suitable for small standard deviations.

B. Timing Attack Vulnerabilities

Side channel attacks are physical attacks, based on information gained from the physical implementation of a cryptosystem. To date, little research has been conducted on the vulnerabilities of LBC implementations to physical attacks; efforts so far are summarised by Hodgers et al. [22]. Timing analysis attacks are highly algorithm-specific in nature, where the dependency between the execution time of an algorithm and its secret internal states is exploited. Attacks on LBC constructions are emerging [23], [5]. The timing-attack countermeasure is to guarantee an execution time independent of the secret values [24]. This can be achieved by ensuring

\(^\dagger\)The same parameters are used for implementing the ring-LWE encryption scheme of Lyubashevsky et al. [16], see [17] for a hardware implementation.
constant response time [21] or subsequent random shuffling of the secret values [7]. The following definition of time independence is used in this research:

**Definition 2 (Time independence):** A program achieves the property of independent-time when no information about the secret value(s) is leaked by the timing of the program.

**IV. Constant-Time CDT Hardware Architectures**

A constant-time implementation of a CDT sampler is achievable by comparing the table in a fixed number of clock cycles. Algorithm 1 shows that an early termination is possible if the comparison of uniformly sampled \( r \) and the \( S[mid] \) returns an equality. This exact match could happen with a small probability of \( 2^{-\lambda}N \). This early termination is avoided by not monitoring an exact match of \( r \) and \( S[mid] \) separately. Hence, the binary search algorithm is always bounded between \([\lceil \log_2(N) \rceil, \lceil \log_2(N) \rceil]\) search iterations of the for loop. Consequently, where \( N \) is a power of two, the algorithm executes exactly in constant-time; and for all other \( N \), the algorithm is tweaked to occasionally perform an extra iteration to ensure the algorithm complexity is fixed to \( \lceil \log_2(N) \rceil \).

**A. Trivium as a PRNG**

The CDT sampler requires uniformly distributed samples, for which Trivium [25] is selected, due to its versatility. It is a synchronous, binary stream cipher with a 288-bit internal state. To achieve a large number of uniformly random bits per clock cycle, the Trivium modules are unrolled. The resources for unrolled designs compared to standard Trivium×1 are rather negligible: 28 additional LUTs, 63 additional flip-flops, and 15 additional slices for Trivium×8 and 26 additional LUTs, 147 additional flip-flops, and 21 additional slices for Trivium×32 on a Spartan-6 LX25-3 device, post place-and-route.

**B. Proposed Constant-Time CDT Sampler For Encryption**

Figure 1a illustrates the proposed constant-time CDT sampler for encryption. The CDF table \( S[r] \) consists of \( N = 32 \) \((r \times \sigma)\) entries, with \( \lambda = 64 \). A single ported ROM, a 5-bit address port and a 64-bit data port are employed. Trivium×64 is used for uniform sample generation, with module initialisation (key setup, IV setup, and the randomisation phase) handled externally at startup, and thereafter controlled by the binary search state-machine, BinSearch. The uniform samples are only generated when required, saving circuit power.

The BinSearch state-machine begins at the SET state, resetting the three pointers \((min, mid, and max)\) to initial values (as in Algorithm 1). It transitions unconditionally to the SEARCH state in the next clock cycle and these three pointers are updated, given the result from the 64-bit comparison. After exactly 5 cycles, a search is found, and the state generates a single bit hit. The Trivium module is activated by this hit to request a new uniformly random 64-bit value. The buffered mid is combined with a random bit \( b \), which attaches a sign to the generated discrete Gaussian sample \( x \).

**C. Proposed Constant-Time CDT Sampler For Signatures**

The proposed CDT sampler for signatures uses two state-machines, BinSearch0 and BinSearch1, to parallelise two independent searches (see Figure 1b). Since most FPGA devices have dual ported BRAMs, the CDF table for both state-machines can be accessed from one BRAM. Each state-machine has an 8-bit address and a 64-bit data port. The two state machines each get a 64-bit uniformly random number, \( r_0 \) and \( r_1 \) respectively, from the PRNG, in two consecutive clock cycles. During the next 8 clock cycles, \( r_0 \) and \( S[mid0] \) are compared in BinSearch0 and the three pointers are updated. Simultaneously, \( r_1 \) is processed at BinSearch1. The state-machines work independently to generate two independent random samples \( x_0 \) and \( x_1 \) in 8 clock cycles; the two samples are then combined as \( x = x_0 + 11x_1 \), where lastly a sign bit is assigned.

**V. RESULTS, EVALUATION AND COMPARISON**

This section presents the post place-and-route performance results and comparison of the proposed samplers with existing sampler implementations for encryption and signatures. Xilinx ISE 14.7 is used, and where possible comparable implementations have been re-run on the same FPGA device in order to fairly compare the results. Throughput and throughput per area (TPAR) have been evaluated for all schemes, in terms of sampling operations per second (Ops/s) and sampling operations per second per slice (Ops/s/S). Table I gives the CDT sampler resource consumption for both encryption and signature parameters. A low cost Spartan-6 FPGA is targeted, and low area and balanced results are presented, where the area-optimised designs employ BRAM, unlike the balanced designs, which offer higher running frequencies.

Table II shows performance results of the samplers, compared with existing CDT hardware samplers. For encryption parameters, a single ported distributed ROM comprising of LUTs is proposed in the design without BRAM. The slice resources can be significantly reduced if BRAMs are utilised. However, the price of resource reduction is paid for by reduced operable frequency. The only other constant-time CDT implementation for encryption is by Poppelmann and Guneysu [21], which generates a single sample per cycle. However, it is 4x slower in frequency with 5x many slices, and thus this research offers a more lightweight, constant-time alternative. The CDT sampler design by Du and Bai [20] is lightweight but it is only for encryption and does not run in independent-time. For signature parameters, the implementation by Poppelmann et al. [19] operates in non-constant time but has a lower throughput per slice than this work. Thus the CDT sampler proposed in this research is preferable for practical implementations.

<table>
<thead>
<tr>
<th>Sampler</th>
<th>Registers</th>
<th>LUT</th>
<th>Slice</th>
<th>BRAM</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) CDT_Area</td>
<td>17</td>
<td>53</td>
<td>16</td>
<td>2</td>
<td>136</td>
</tr>
<tr>
<td>b) CDT_Balance</td>
<td>26</td>
<td>66</td>
<td>21</td>
<td>0</td>
<td>394</td>
</tr>
<tr>
<td>b) CDT_Area</td>
<td>48</td>
<td>130</td>
<td>44</td>
<td>2</td>
<td>126</td>
</tr>
<tr>
<td>c) CDT_Balance</td>
<td>64</td>
<td>577</td>
<td>197</td>
<td>0</td>
<td>130</td>
</tr>
<tr>
<td>Available</td>
<td>30,064</td>
<td>24,051</td>
<td>3,758</td>
<td>52</td>
<td>-</td>
</tr>
</tbody>
</table>
VI. CONCLUSION

In this research two independent-time hardware designs of a discrete Gaussian CDT sampler are proposed, suitable for encryption and signature applications, with a focus on low-area footprint and high throughput. Resistance against timing attacks is achieved by ensuring constant execution time. Moreover, the proposed hardware CDT sampler designs clearly outperform the previously proposed samplers.

REFERENCES


