Polar Sampler: A Novel Bernoulli Sampler Using Polar Codes with Application to Integer Gaussian Sampling

Jiabo Wang[1] and Cong Ling[2]

[1] Tsinghua University, Beijing 100084, China
wangjiabo@mail.tsinghua.edu.cn
[2] Imperial College London, London SW7 2AZ, UK
c.ling@imperial.ac.uk

Abstract. Cryptographic constructions based on hard lattice problems have emerged as a front runner for the standardization of post quantum public key cryptography. As the standardization process takes place, optimizing specific parts of proposed schemes, e.g., Bernoulli sampling and Integer Gaussian sampling, becomes a worthwhile endeavor. In this work, we propose a novel Bernoulli sampler based on polar codes, dubbed "polar sampler". The polar sampler is information theoretically optimum in the sense that the number of uniformly random bits it consumes approaches the entropy bound asymptotically. It also features quasi-linear complexity and constant-time implementation. An integer Gaussian sampler is developed using multilevel polar samplers. Our algorithm becomes effective when sufficiently many samples are required at each query to the sampler. Security analysis is given based on Kullback-Leibler divergence and Rényi divergence. Experimental and asymptotic comparisons between our integer Gaussian sampler and state-of-the-art samplers verify its efficiency in terms of entropy consumption, running time and memory cost. We envisage that the proposed Bernoulli sampler can find other applications in cryptography in addition to Gaussian sampling.

Keywords: Bernoulli sampling · integer Gaussian sampling · Polar codes · Integer lattice · Kullback-Leibler divergence · Rényi divergence · Constant-time.

1 Introduction

Lattice-based cryptography is one of the most promising candidates of cryptosystems in the plausible post-quantum age. The security of lattice-based primitives is guaranteed by the hardness of worst-case lattice problems, e.g. the Learning With Errors (LWE) problem [29, 18] and Short Integer Solution (SIS) problem [21, 20]. The discrete Gaussian distribution lies at the core of security proofs of these primitives, and it is also one of the fundamental building blocks of practical lattice-based cryptographic applications, e.g. signature schemes, encryption and key exchanges. In general, the security level of these cryptographic applications is closely related to the statistical performance of the discrete Gaussian

sampling (DGS) algorithm. From an implementation standpoint, cryptographers also take other qualities of a DGS into consideration including side-channel resistance, computation and storage efficiency. In practice, the tradeoff between these performances is a bottleneck of this problem.

It has been widely assumed that for cryptographic applications with λ bits of security the statistical distance (SD) between the ideal distribution and the approximated one should be roughly $2^{-\lambda}$ such that there is only minor loss in security [10]. Some other measures such as Kullback-Leibler (KL) divergence and Rényi divergence are proved to provide more efficient security analysis than SD, as they can lower the requirement for precision and reduce the cost of the algorithms in many practical cases [26–28, 3]. From a practical point of view, the difficulty of DGS lies in the implementation of DGS in cryptographic primitives with constrained resources. Besides the resilience against potential side-channel attacks, a designer looking for the optimal DGS solution to a specific application strikes the balance of memory consumption and running time, precision and efficiency.

There are already a variety of works addressing the application of DGS in lattice-based primitives. Existing techniques include the binary sampling [9], the cumulative distribution table (CDT) sampler [8], the Knuth-Yao sampler [17], and the discrete Ziggurat sampler [19], etc. In [12], rejection sampling is used to generate discrete Gaussian samples where one draws an element x from a discrete domain uniformly at random and accepts it with probability proportional to $\exp(-x^2/2\sigma^2)$ where σ is the standard deviation. However, calculating the exponential function requires high-precision computing and sufficient trials are needed before the sampler produces an output. In [9], binary search through CDT is used in the signature scheme BLISS. At the first step of BLISS, a discrete Gaussian vector is generated to blind the secret. However, the CDT sampling itself takes 35 percent of the total running time of BLISS [16] and the precomputed CDT requires larger memory especially when a wider distribution is in need to improve the security level.

In [15], Hülsing et al. replaced the discrete Gaussian distribution by a rounded Gaussian distribution in Lyubashevsky's signature scheme without trapdoors and BLISS showing its effectiveness, security and efficiency. As the term suggested, a rounded Gaussian distribution is derived by rounding continuous Gaussian samples which can be efficiently realized by Box-Muller transform [5] in constant time. A convolution method, first proposed in [24], can expand a discrete Gaussian distribution with a small parameter to a wider one. Another sampling design [22] exploits a base sampler with small parameters to efficiently generate DGS with arbitrary and varying parameters in a convolutional manner. This application-independent algorithm consists of an online and offline stage, both of which can be carried out in constant time, proving a resilience against timing attack. A constant-time sampler was proposed in [34] and Rényi divergence was used to improve the efficiency. In [30], arithmetic coding, a classical data compression technique, was adopted as a sampler in BLISS giving a reduced signature size.

When reviewing the literature of DGS, we find that Bernoulli sampling is of vital importance to randomness generation. It is involved in many cryptographic designs and a typical example is BLISS [9] where Bernoulli sampling is employed to build a discrete Gaussian sampler. To make BLISS safe under side channel attacks, especially the timing-based one, improved Bernoulli samplers were devised in [6, 25, 11, 34]. To improve the efficiency of Bernoulli sampling with biases in exponential or cosh form, as is the case in BLISS, polynomial approximation with sufficient precision were proposed in [34, 4]. Our research begins with Bernoulli sampling and we get our inspiration from polar source coding. The proposed Bernoulli sampler can be used for generating arbitrary discrete distribution and this paper is concerned about its application to Gaussian sampling.

Contribution In this work, we propose a novel Bernoulli sampler using polar codes and apply it to DGS over the integers. Polar codes are the first class of efficiently encodable and decodable codes which provably achieve channel capacity of symmetric channels [2]. It can also achieve Shannon's data compression rate [1]. The power of polar codes stems from the polarization phenomenon: under Arıkan's polar transform, information measures of synthesized sources (or channels) converge to either 0 or 1 when coding becomes trivial. Moreover, the state-of-the-art decoding runs with $O(N \log \log N)$ complexity where N denotes the block length of a polar code [33]. Given their attractive performance, polar codes have found a wide range of applications in information theory and communication systems. In particular, they have been standardized for the upcoming fifth-generation (5G) wireless communication networks.

This work tackles the sampling problem from a source coding perspective, namely, sampling can be considered the inverse problem of source coding. In source coding or data compression, one typically encodes a block of symbols of a certain distribution into some bits which become uniformly random as the block length tends to infinity [7]. Since a source code is invertible, inverting this process would produce samples from the desired distribution. When a large number of independent Gaussian samples are required in cryptographic applications (e.g. fully homomorphic encryption (FHE), digital signatures), polar sampling is well suited because in this case the information source of distribution $D_{\mathbb{Z}^N,c,s}$ is memoryless. Note that the polar sampler is not restricted to sampling from the discrete Gaussian distribution, but can be extended to other distributions of interest in cryptography.

The principal contributions of this paper are summarized as follows:

- A novel approach to sample from a Bernoulli distribution as well as an integer Gaussian sampler using multilevel polar samplers are developed. Using a binary partition tree, we recursively partition $\mathbb{Z}$ into 2 cosets, 4 cosets, and so on. The number of partitions is only logarithmic in s. Each partition gives rise to a binary source, which is produced by one polar sampler. The advantage of this multilevel sampling approach is that only Bernoulli samples are needed, which allow simpler implementation than sampling over the whole integer domain.

- Analysis of approximation errors. Although multilevel polar samplers would produce the desired distribution $D_{\mathbb{Z}^N,c,s}$, it is not exactly so. This is because the polar sampler converts N i.i.d. Bernoullis into N polarized and unpolarized Bernoullis. We approximate the polarized ones using either unbiased or determinisitic Bernoullis which will only yield an approximate version of the desired distribution. We derive upper bounds on the closeness between the target discrete Gaussian and its approximation measured by KL divergence.
- Security analysis. To achieve a certain security level in a standard cryptographic scheme with oracle access to a discrete Gaussian distribution, the principle of setting the parameters of our polar sampler is also discussed based on KL divergence. In cryptographic applications where the number of queries q to the Gaussian sampler is limited (e.g., $q \leq 2^{64}$ in the NIST specifications of signatures), using Rényi divergence can yield considerable savings according to previous work of [28, 3]. We also apply Rényi divergence to improve the parameter selection of polar sampler.

The proposed multilevel polar sampler scheme complements and distinguishes from existing discrete Gaussian samplers in the literature. In addition to offering a different approach, it exhibits several salient features:

- Information theoretic optimality. Asymptotically, the multilevel polar sampler achieves the entropy bound of the discrete Gaussian distribution. This implies that it requires minimum resources of random bits to produce the desired distribution.
- Quasi-linear complexity. The proposed Gaussian sampling approach enjoys low complexity. The design of a polar sampler can be done at the offline stage, that is, given a target distribution, it is done once and for all. The online stage of a polar sampler computes certain posteriori probabilities which can be implemented in $O(N \log N)$ complexity[3]. We also give experimental and asymptotic comparison between our DGS approach and other exising samplers including Knuth-Yao sampling, binary sampling and CDT sampling. The prominent advantage of multilevel polar sampler is the entropy consumption which indicates the cost of randomness. The overall running time depends on both SC decoding(computing LRs) and Bernoulli sampling. Compared with the binary sampling [9], polar sampler has higher computational complexity but it asymptotically and effectively reduces the entropy consumption. We also illustrate in experiments that the multilevel polar sampler is faster than Knuth-Yao sampling.
- Constant-time implementation. The multilevel polar sampler also admits constant-time implementation, since a polar code has a fixed-length. This compares favorably with other source coding techniques such as Huffman coding which has varying codeword length. Moreover, all the operations of the proposed sampler run in constant time. This makes our sampler very attractive when dealing with timing side-channel attacks.

[3] It can be upgraded to $O(N \log \log N)$ using the state-of-the-art SC decoding [33].

Of course, the proposed sampler can be combined with existing "expander" techniques such as convolution if needed. In this work, we focus on the theoretic design and analysis of polar samplers, whereas various optimization issues (e.g., concrete computational/storage costs, finite precision etc.) are left to future work. Nevertheless, we have found it in experiments that even a prototype implementation significantly outperforms the Knuth-Yao sampler in speed in benchmark experiments.

Roadmap The roadmap of this paper is given as follows. Section 2 introduces the proposed Bernoulli sampler, i.e., polar sampler, and elucidates its relation to polar source coding. Section 3 presents how we devise an integer Gaussian sampler using multiple polar samplers. Section 4 analyses the approximation error of our integer Gaussian sampler based on KL divergence. In Section 5, the security, precision and parameter selection are discussed based on KL and Rényi divergence. Section 6 compares our integer Gaussian sampler with state-of-the-art samplers regarding the complexity. Section 7 concludes this paper.

2 Bernoulli Sampling Using Polar Codes

2.1 Notation

Given a vector $x^{1:N}$ and a set $\mathcal{A} \subset \{1, \cdots, N\}$, $x_{\mathcal{A}}$ denotes the subvector of $x^{1:N}$ indexed by $\mathcal{A}$. Denote by $X \sim P$ a distribution P of X over a countable set $\mathcal{X}$. Then the entropy of X is defined as $H_P(X) = -\sum_{x\in\mathcal{X}} p(x) \log p(x)$. We write $H(X) = H_P(X)$ for brevity if the distribution is clear. Suppose X and Y have a joint distribution $P(X, Y)$. The conditional entropy of X given Y is defined as $H(X|Y) = \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} p(x, y) \log \frac{p(y)}{p(x,y)}$. The logarithm to base 2 is denoted by log while the natural logarithm is denoted by ln.

2.2 Source Polarization

The key idea of polar source coding can be found in [1] where a polar code was proposed to achieve Shannon's source coding bound. Let $(X^{1:N}, Y^{1:N})$ denotes N i.i.d. copies of a memoryless source (X, Y) of joint distribution $P_{X,Y}$, where X takes values over $\mathcal{X} = \{0, 1\}$ while Y takes values over a countable set $\mathcal{Y}$. The two random source X and Y are correlated, and Y is called the side-information[4]. In source coding, the encoder compresses a sequence $X^{1:N}$ into a shorter codeword, such that the decoder can produce an estimation $\hat{X}^{1:N}$ of $X^{1:N}$ using the codeword side information $Y^{1:N}$.

Polar codes are proved to achieve Shannon's source coding bound asymptotically. The source polarization transform from $X^{1:N}$ to $U^{1:N}$ is performed by

[4] Note that even if there is only one source in the context of this paper, the proposed multilevel code still needs side information (which is basically information coming from lower levels.)

applying an entropy-preserving circuit to $X^{1:N}$, i.e.,

$$U^{1:N} = X^{1:N}G_N,\ G_N = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{\otimes n} B_N,$$

where $^{\otimes n}$ denotes the n-th Kronecker power, and B_N is a bit-reversal permutation [2] of the input vector. Fig. 1 illustrates the source polarization transforms of $X^{1:2}$ and $X^{1:4}$ where $\oplus$ denotes mod-2 sum. This transform preserves the entropy in the sense that

$$H(U^{1:2} \mid Y^{1:2}) = 2H(X \mid Y),\ H(U^{1:4} \mid Y^{1:4}) = 4H(X \mid Y).$$

Meanwhile, it also polarizes the entropy in the sense that

$$H(U^{(1)} \mid Y^{1:4}) \geq H(S^{(1)} \mid Y^{1:2}) = H(S^{(2)} \mid Y^{3:4}) \geq H(U^{(2)} \mid Y^{1:4}, U^{(1)}),$$

and

$$\begin{aligned} H(U^{(3)} \mid Y^{1:4}, U^{1:2}) \geq & H(R^{(1)} \mid Y^{1:2}, S^{(1)}) \\ = & H(R^{(2)} \mid Y^{3:4}, S^{(2)}) \geq H(U^{(4)} \mid Y^{1:4}, U^{1:3}). \end{aligned}$$

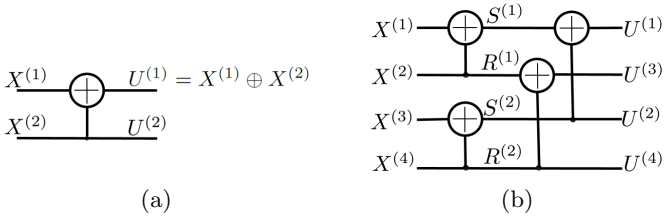

Fig. 1. The source polarization transform [2]: (a) A two-by-two transform (b) A four-by-four transform.

By applying the construction in Fig. 1 recursively, we derive a bijection $U^{1:N} = X^{1:N}G_N$ inducing a combined source pair $(U^{1:N}, Y^{1:N})$ and a transition $W_N : U^{1:N} \to Y^{1:N}$. This combined source pair is then split into N synthesized source pairs $(U^{(i)}, Y^{1:N} \times U^{1:i-1})$ giving rise to N sub-transitions $W_N^{(i)} : U^{(i)} \to Y^{1:N} \times X^{1:i-1}$. A polarization phenomenon happened to sub-source pairs is observed and stated as follows.

Theorem 1 (Source Polarization [1]). *Let (X, Y) be a source as above. For any $N = 2^n$,$n \geq 1$, let $U^{1:N} = X^{1:N}G_N$. Then, for any $0 < \beta < 0.5$, as $N \to \infty$,*

$$\frac{\left|\left\{i \in [1, N] : H(U^{(i)} \mid Y^{1:N}, U^{1:i-1}) \in (1 - 2^{-N^\beta}, 1]\right\}\right|}{N} \to H(X \mid Y) \qquad (1)$$

$$\frac{\left|\left\{i \in [1, N] : H(U^{(i)} \mid Y^{1:N}, U^{1:i-1}) \in [0, 2^{-N^\beta})\right\}\right|}{N} \to 1 - H(X \mid Y). \qquad (2)$$

Note that in the absence of side information Y, the above theorem still holds by considering Y independent of X.

Definition 1 (Bhattacharyya Parameter [13]). *Let $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ be a pair of random variables where $\mathcal{X} = \{0, 1\} = GF(2)$ and $\mathcal{Y}$ is an arbitrary finite set. Let X and Y follow the joint distribution $P_{XY}(x, y)$. If X is the source to be compressed and Y is the side information, the Bhattacharyya parameter is defined as*

$$\begin{aligned} Z(X|Y) &\equiv 2\sum_{y} P_Y(y)\sqrt{P_{X|Y}(0|y)P_{X|Y}(1|y)} \\ &= 2\sum_{y} \sqrt{P_{X,Y}(0,y)P_{X,Y}(1,y)}. \end{aligned} \tag{3}$$

Proposition 1 ([1], Proposition 2).

$$(Z(X|Y))^2 \leq H(X|Y) \tag{4}$$

$$H(X|Y) \leq \log(1 + Z(X|Y)). \tag{5}$$

It is implied by Proposition 1 that for a source (X, Y), the parameters $H(U^{(i)} \mid Y^{1:N}, U^{1:i-1})$ and $Z(U^{(i)} \mid Y^{1:N}, U^{1:i-1})$ polarize simultaneously in the sense that $H(U^{(i)} \mid Y^{1:N}, U^{1:i-1})$ approaches 0 (resp. 1) as $Z(U^{(i)} \mid Y^{1:N}, U^{1:i-1})$ approaches 0 (resp. 1).

For $\beta \in (0, 1/2)$ and $\alpha = 2^{-N^\beta}$, the indexes of $U^{1:N}$ can be divided into a low-entropy set

$$\mathcal{L}_{X|Y} = \left\{ i \in [N] : \ Z(U^{(i)} \mid Y^{1:N}, U^{1:i-1}) \in [0, \alpha) \right\} \tag{6}$$

and its complement ${\mathcal{L}^c}_{X|Y}$. Again, in the absence of side information Y, the two sets are defined in the same way by considering Y independent of X and U.

This gives rise to the encoding and decoding scheme described in [1]. More specifically, for a realization of $(X^{1:N}, Y^{1:N}) = (x^{1:N}, y^{1:N})$, the encoder computes $u^{1:N} = x^{1:N} G_N$ and only shares $u_{\mathcal{L}^c_{X|Y}}$ with the decoder. The compression rate is defined as $R = |\mathcal{L}^c_{X|Y}|/N$. The decoder can obtain an estimate $\hat{u}^{1:N}$ of $u^{1:N}$ in a successive manner as

$$\hat{u}^{(i)} = \begin{cases} u^{(i)}, & \text{if } \ i \in \mathcal{L}^c_{X|Y} \\ 0, & \text{if } \ i \in \mathcal{L}_{X|Y} \text{ and } L_N^{(i)}(y^{1:N}, \hat{u}^{1:i-1}) \geq 1 \\ 1, & \text{if } \ i \in \mathcal{L}_{X|Y} \text{ and } L_N^{(i)}(y^{1:N}, \hat{u}^{1:i-1}) < 1, \end{cases} \tag{7}$$

where $L_N^{(i)}(y^{1:N}, \hat{u}^{1:i-1})$ is called the likelihood ratio (LR) defined by

$$L_N^{(i)}(y^{1:N}, \hat{u}^{1:i-1}) = \frac{P(U^{(i)} = 0 | Y^{1:N} = y^{1:N}, U^{1:i-1} = \hat{u}^{1:i-1})}{P(U^{(i)} = 1 | Y^{1:N} = y^{1:N}, U^{1:i-1} = \hat{u}^{1:i-1})}. \tag{8}$$

Theorem 2 (An upper bound on error probability [1]). *For any fixed* $R > H(X|Y)$ *and* $\beta < 0.5$*, the probability of error for the above polar source coding method is bounded as* $P_e = \Pr(\hat{U}^{1:N} \neq U^{1:N}) = O(2^{-N^\beta})$.

It implies that any rate $R > H(X|Y)$ is achievable with a vanishing block error probability for sufficiently large N. As N goes to infinity, the polarization process removes the randomness of the low-entropy set almost surely while the other set becomes random. Additionally, the complexity of polar encoding and decoding are both $O(N \log N)$.

2.3 From Source Coding to Bernoulli Sampling

Recall the memoryless source $(X, Y) \sim P_{X,Y}$ and polar source coding techniques we introduced in Section 2.2. Now consider the sampling problem, i.e., to sample from a Bernoulli distribution $P(X)$ given (or without) side information Y. We propose a novel Bernoulli sampler called PolarSampler$(\cdot)$ in Algorithm 1. The interfaces, key operations and subroutines are described as follows.

input : $N = 2^n$,$\mathbf{c}[y^{1:N}]$,$\mathcal{H}_{X|Y}$, $\mathcal{L}_{X|Y}$ (Or $\mathbf{c}[1^{1:N}] = [c_1, \cdots, c_1]$, $\mathcal{H}_X, \mathcal{L}_X$ without Y.)
output: $x^{1:N}$

Define global arrays:LRReg $[N][n+1]$, UReg $[N][n+1]$;
LRReg $[:][0]=(1 - \mathbf{c}[y^{1:N}])/\mathbf{c}[y^{1:N}]$;
for $i \leftarrow 1$ **to** N **do**
    LRReg $\leftarrow$ `CalLR(n,i)`;
    if *index* $i \in \mathcal{H}_{X|Y}$ **then**
        UReg$[i][n] \leftarrow$`randomBin(); // equation(10).`
    end
    else if *index* $i \in \mathcal{L}_{X|Y}$ **then**
        UReg$[i][n] =$ LRReg$[i][n] < 1$; `// equation (10).`
    end
    else index $i \in \mathcal{H}^c_{X|Y} \backslash \mathcal{L}_{X|Y}$
        UReg$[i][n] =$`Uniform()` $< 1/(1 +$ LRReg$[i][n])$; `// Uniform() produces values in (0,1] uniformly at random; equation (11).`
    end
    UReg $\leftarrow$ `CalBit(n,i)`;
end
return $x^{1:N} =$ UReg$[:][0] \cdot G_N$

Algorithm 1: PolarSampler$(\cdot)$.

Interfaces PolarSampler$(\cdot)$ draws N samples $x^{1:N}$ from the target distribution $P(X)$ given N samples $y^{1:N}$ of the side information Y. It has access to a pre-computed table $\mathbf{c} = [c_1, \cdots, c_{|\mathcal{Y}|}]$ where each element indicates a Bernoulli bias

$c_p = P(X = 1|Y = y_p)$ and $y_p \in \mathcal{Y}$. The input vector $\mathbf{c}[y^{1:N}]$ is defined as $[c_{y^{(1)}}, \cdots, c_{y^{(N)}}]$ where $c_{y^{(i)}} = c_p$ if $y^{(i)} = y_p$.

In addtion to the low-entropy set $\mathcal{L}_{X|Y}$ as defined in formula (6), we further define a high-entropy set[5]

$$\mathcal{H}_{X|Y} = \left\{ i \in [N] : \ Z(U^{(i)} \mid Y^{1:N}, U^{1:i-1}) \in (1-\alpha, 1] \right\}, \tag{9}$$

where $\alpha = 2^{-N^\beta}$. In order to define the two sets $\mathcal{H}_{X|Y}$ and $\mathcal{L}_{X|Y}$, one needs to calculate the Bhattacharyya parameter $Z(U^{(i)}|Y^{1:N}, U^{1:i-1})$ efficiently. However, as a source pair (X, Y) turns into a synthesized source pair $(U^{(i)}, Y^{1:N} \times U^{1:i-1})$ by polarization transform, the alphabet size of the side information increase exponentially with N. Calculating $Z(U^{(i)}|Y^{1:N}, U^{1:i-1})$ according to how we define it in Definition 1 becomes intractable. Some efficient algorithms to calculate the Bhattacharyya parameters were proposed in [32][6]. In addition, preparing the bias table $\mathbf{c}$ and calculating $Z(U^{(i)}|Y^{1:N}, U^{1:i-1})$ are done offline. We will refer to the offline stage as the construction stage of PolarSampler$(\cdot)$ in the sequel.

Note that when there is no side information Y, the precomputed table $\mathbf{c}$ only consists of one bias $c_1 = P(X = 1)$. The input bias vector will be $\mathbf{c}[1^{1:N}] = [c_1, \cdots, c_1]$ of length N. The high and low entropy sets $\mathcal{H}_X$ and $\mathcal{L}_X$ are defined in the same manner by considering Y independent of X.

Key Operations According to polar source coding, the $U^{(i)}$ for $i \in \mathcal{H}_{X|Y}$ with very high entropy is approximately uniformly distributed and is approximately independent of both $U^{1:i-1}$ and $Y^{1:N}$, while the $U^{(i)}$ for $i \in \mathcal{L}_{X|Y}$ with very low entropy is almost deterministic. Those $U^{(i)}$ for $i \in \mathcal{H}^c_{X|Y} \backslash \mathcal{L}_{X|Y}$ (i.e., $Z(U^{(i)} \mid Y^{1:N}, U^{1:i-1}) \in [\alpha, 1-\alpha]$) are unpolarized. As N goes to infinity, the fraction of unpolarized indexes vanishes and the fraction of high-entropy indexes approaches the entropy $H(X|Y)$ of X given Y according to Theorem 1.

Since the polarization transform G_N is reversible, it is expected to produce an approximation $Q_{X^{1:N}}$ of $P_{X^{1:N}}$ by applying the above transform to $U^{1:N}$, i.e. $X^{1:N} = U^{1:N} G_N$. However, the unpolarized set may not be negligible for finite length N, and should be handled with care. More precisely, those $U^{(i)}$ for $i \in \mathcal{H}^c_{X|Y} \backslash \mathcal{L}_{X|Y}$ are not quite uniform; feeding them with uniform bits may cause non-negligible distortion from the target distribution. Therefore, for sampling, $U^{(i)}$s should follow the distribution:

$$U^{(i)} = \begin{cases} \{0,1\} \sim \text{Bernoulli}(0.5), & \text{if } i \in \mathcal{H}_{X|Y} \\ \arg\max_u P_{U^{(i)}|Y^{1:N}, U^{1:i-1}}(u|y^{1:N}, u^{1:i-1}), & \text{if } i \in \mathcal{L}_{X|Y} \end{cases} \tag{10}$$

[5] In [1], $\mathcal{L}^c{}_{X|Y}$ is called the high-entropy set, which is larger than $\mathcal{H}_{X|Y}$.

[6] Matlab codes available in the folder .../PolarFastSCL/HowToConstructPolarCode at https://github.com/YuYongRun/PolarCodeDecodersInMatlab.git.

and

$$U^{(i)} = \begin{cases} 0 & w.p.\ P_{U^{(i)}|Y^{1:N},U^{1:i-1}}(0|y^{1:N},u^{1:i-1}) \\ 1 & w.p.\ P_{U^{(i)}|Y^{1:N},U^{1:i-1}}(1|y^{1:N},u^{1:i-1}) \end{cases} \text{if } i \in \mathcal{H}^c_{X|Y}\backslash\mathcal{L}_{X|Y} \quad . \tag{11}$$

Fig. 2 shows the difference between source coding and sampling: although the unpolarized set belongs to the compressed codeword in source coding, its bits should be randomized as in (11) in sampling.

While sampling according to formula (10) is obviously trivial and straightforward, the bottleneck of entropy consumption is determined by sampling the unpolarized set in formula (11). The following lemma is adpated from [23, Lemma 1] which gives the fraction of unpolarized set for finite $n = \log N$.

Lemma 1 (Fraction of unpolarized set [23]). *Let μ ($3.579 \leq \mu \leq 4.714$) be a constant called the upper bound on the scaling exponent which is solely determined by the conditional probability $P_{X|Y}$. For a constant $v > 1$ and $n = \log N \geq 1$, the following relation holds:*

$$\frac{\left|\left\{i \in [1,N] : Z(U^{(i)} \mid Y^{1:N}, U^{1:i-1}) \in [2^{-vn},\ 1-2^{-vn}]\right\}\right|}{N} < c2^{-n/\mu},$$

where the constant c depends solely on v and it does not depend on n or $P_{X|Y}$.

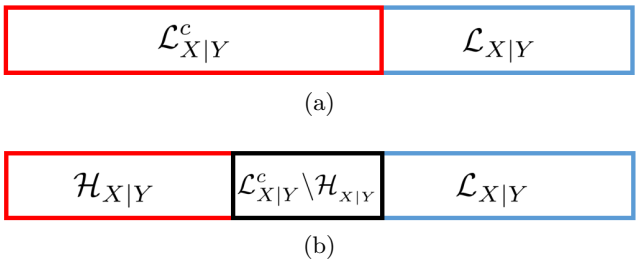

Fig. 2. Polar source coding vs. polar sampling: (a) Subsets of indexes for polar source coding (b) Subsets of indexes for polar sampling. The fraction of $\mathcal{L}^c_{X|Y}\backslash\mathcal{H}_{X|Y}$ vanishes as N goes to infinity.

Corollary 1 (Asymptotic property of polarization). *Let $X \sim P(X)$ be the target Bernoulli distribution with side information Y. As $N \to \infty$, the fraction of $\mathcal{H}_{X|Y}$ goes to $H(X|Y)$, the fraction of $\mathcal{L}_{X|Y}$ goes to $1 - H(X|Y)$ and the fraction of $\mathcal{L}^c_{X|Y}\backslash\mathcal{H}_{X|Y}$ scales as $N^{-\frac{1}{\mu}}$ for $3.579 \leq \mu \leq 4.714$.*

In the absence of side information Y, the above property also holds by substituting X for $X|Y$.

Proof. Recall it from Proposition 1 that $Z(U^{(i)} \mid Y^{1:N}, U^{1:i-1})$ and $H(U^{(i)} \mid Y^{1:N}, U^{1:i-1})$ polarize simultaneously. Therefore Theorem 1 can be restated as follows. For any $0 < \beta < 0.5$, as $N \to \infty$,

$$\frac{\left|\mathcal{H}_{X|Y} \coloneqq \left\{i \in [1, N] : Z(U^{(i)} \mid Y^{1:N}, U^{1:i-1}) \in (1 - 2^{-N^\beta}, 1]\right\}\right|}{N} \to H(X \mid Y)$$

$$\frac{\left|\mathcal{L}_{X|Y} \coloneqq \left\{i \in [1, N] : Z(U^{(i)} \mid Y^{1:N}, U^{1:i-1}) \in [0, 2^{-N^\beta})\right\}\right|}{N} \to 1 - H(X \mid Y).$$

Given a threshold 2^{-N^β}, we can find a v such that $N^{-v} = 2^{-N^\beta}$. Lemma 1 implies that the Bhattacharyya parameter $Z(U^{(i)} \mid Y^{1:N}, U^{1:i-1})$ falls on the interval $[N^{-v}, 1 - N^{-v}]$ with a probability smaller than $cN^{-1/\mu}$ where $3.579 \leq \mu \leq 4.714$ and c is determined by v and $P(X|Y)$. Therefore, the fraction of unpolarized set $\mathcal{L}^c_{X|Y} \backslash \mathcal{H}_{X|Y}$ scales as $N^{-1/\mu}$. The above conclusions still hold in the absence of side information Y by considering X independent of Y. Q.E.D.

Subroutines To carry out the operations in formula (10) and (11), one also needs to calculate $P_{U^{(i)}|Y^{1:N},U^{1:i-1}}$. Recall that the definition of likelihood ratio is

$$L_N^{(i)}(y_1^N, u^{1:i-1}) = \frac{P_{U^{(i)}|Y^{1:N},U^{1:i-1}}(0|y^{1:N}, u^{1:i-1})}{P_{U^{(i)}|Y^{1:N},U^{1:i-1}}(1|y^{1:N}, u^{1:i-1})} \tag{12}$$

Since these likelihood ratios can be computed with quasi-linear complexity $O(N \log N)$ by SC decoding proposed in [2], the posterior probability $P_{U^{(i)}|Y^{1:N},U^{1:i-1}}$ in (10) and (11) can be equivalently computed. When there is no side information Y, we can compute $P_{U^{(i)}|U^{1:i-1}}$ in the same manner as above by considering Y independent of X.

In PolarSampler$(\cdot)$, we first define a two-dimensional likelihood ratio array LRReg$[N][n+1]$ and a two-dimensional bit array UReg$[N][n+1]$ indexed by integers $1 \leq i \leq N$ and $0 \leq m \leq n$. We also define an arrray of $N \times (n+1)$ nodes connected by multiple 2-by-2 butterfly circuits "$\boxtimes$" as in Fig. 3 for $N = 8$. Each node takes the responsibility to update a unique element of LRReg$[N][n+1]$ and a unique element of UReg$[N][n+1]$ of the same index. We define two properties of each layer of the array, i.e., phase and branch denoted by integers ϕ and ψ, respectively. In Fig. 3, we distinguish different phases at each layer by different colors. At layer m, $1 \leq \phi \leq 2^m$ and $0 \leq \psi < 2^{n-m}$. Note that for any layer m, each index $1 \leq i \leq 2^n$ has a unique representation as

$$i = \langle \phi, \psi \rangle_m = \phi + 2^m \cdot \psi.$$

And for a generic array A we abbreviate $A[\langle \phi, \psi \rangle_m][m]$ as $A[\langle \phi, \psi \rangle][m]$. To ease the notations, we also use the notation of likelihood ratio $L_{2^m}^{(\phi)}$ to denote the node by which it is calculated. In Line 2 of PolarSampler$(\cdot)$, the raw likelihood ratios $L_1^{(1)} = P(X = 0|y)/P(X = 1|y)$ are stored in LRReg$[:][0]$ given side

information samples $y^{1:N}$. Other elements of LRReg$[i][m]$ = LRReg$[\langle\phi,\psi\rangle][m]$ and UReg$[i][m]$ = UReg$[\langle\phi,\psi\rangle][m]$ for $m > 1$ will be uniquely calculated by node $L_{2^m}^{(\phi)}$ of phase ϕ and branch ψ at layer m. After PolarSampler$(\cdot)$ finishes its work, the likelihood ratios in formula (12) are finally stored in the n-th column LRReg$[:][n]$ of LRReg and the bit vector $U^{1:N}$ is finally stored in UReg$[:][n]$.

The subroutines CalLR$(\cdot)$ and CalBit$(\cdot)$ are employed to recursively calculate and update the likelihood ratio array LRReg$[N][n+1]$ and the bit array UReg[N][n+1]. This process is exactly what SC decoding does as proposed in [2, Section VIII] and modularized in [31, Section II]. As a high level description,

input : m, ϕ
output: updated LRReg

if $m = 0$ **then return**;
set $\kappa \leftarrow \lceil \phi/2 \rceil$;
if $\phi \mod 2 = 1$ **then** `CalLR` $(m-1, \kappa)$;
for $\psi = 0, \cdots, 2^{n-m} - 1$ **do**
  if $\phi \mod 2 = 1$ **then** LRReg $[\langle\phi,\psi\rangle][m] \xleftarrow{\text{equation(13)}}$ (LRReg$[\langle\kappa, 2\psi\rangle][m-1]$, LRReg$[\langle\kappa, 2\psi+1\rangle][m-1]$);
  else temp=UReg $[\langle\phi-1,\psi\rangle][m]$; LRReg $[\langle\phi,\psi\rangle][m] \xleftarrow{\text{equation(14)}}$ (LRReg$[\langle\kappa, 2\psi\rangle][m-1]$, LRReg$[\langle\kappa, 2\psi+1\rangle][m-1]$);
end

Algorithm 2: The CalLR( ) function.

input : m, ϕ
output: updated UReg

if $\phi \mod 2 = 1$ **then return**;
set $\kappa \leftarrow \lceil \phi/2 \rceil$;
for $\psi = 0, \cdots, 2^{n-m} - 1$ **do**
  UReg $[\langle\kappa, 2\psi\rangle][m-1] \leftarrow$ UReg$[\langle\phi-1,\psi\rangle][m] \oplus$ UReg$[\langle\phi,\psi\rangle][m]$;
  UReg $[\langle\kappa, 2\psi+1\rangle][m-1] \leftarrow$ UReg$[\langle\phi,\psi\rangle][m]$;
end
if $\kappa \mod 2 = 0$ **then** `CalBit` $(m-1, \kappa)$;

Algorithm 3: The CalBit( ) function.

CalLR$(\cdot)$ recursively assembles two likelihood ratios of the same phase but different branches at layer $m-1$ (two nodes on RHS of "$\boxtimes$") and derive two new likelihood ratios of different phases but the same branch at layer m (two nodes

on LHS of "⧖") according to formula (13) and (14) as[7]

$$\mathrm{LRReg}[\langle 2\kappa-1,\psi\rangle][m]=\frac{\mathrm{LRReg}[\langle\kappa,2\psi\rangle][m-1]\cdot\mathrm{LRReg}[\langle\kappa,2\psi+1\rangle][m-1]+1}{\mathrm{LRReg}[\langle\kappa,2\psi\rangle][m-1]+\mathrm{LRReg}[\langle\kappa,2\psi+1\rangle][m-1]} \tag{13}$$

$$\mathrm{LRReg}[\langle 2\kappa,\psi\rangle][m]=[\mathrm{LRReg}[\langle\kappa,2\psi\rangle][m-1]]^{1-2\cdot\mathrm{temp}}\cdot\mathrm{LRReg}[\langle\kappa,2\psi+1\rangle][m-1], \tag{14}$$

where $\mathrm{temp}=\mathrm{UReg}[\langle 2\kappa-1,\psi\rangle][m]$. CalBit$(\cdot)$ works in the other way around as in Line 4 and 5 of Algorithm 3 which gives two bits $\mathrm{UReg}[\langle\kappa,2\psi\rangle][m-1]$ and $\mathrm{UReg}[\langle\kappa,2\psi+1\rangle][m-1]$ of the same phase but different branches at layer $m-1$ (two nodes on RHS of "⧖") given two bits $\mathrm{UReg}[\langle 2\kappa-1,\psi\rangle][m]$ and $\mathrm{UReg}[\langle 2\kappa,\psi\rangle][m]$ of different phases but the same branch at layer m (two nodes on LHS of "⧖").

We give an example to demonstrate how CalLR$(\cdot)$ and CalBit$(\cdot)$ work in Figure 3. In PolarSampler$(\cdot)$, CalLR(n,i) is first used for $i=1$ which indicates the node $L_8^{(1)}$. It in turn activates two $L_4^{(1)}$ nodes at layer 2, then four $L_2^{(1)}$ nodes at layer 1 and terminates at eight $L_1^{(1)}$ nodes at layer 0. The nodes at layer 0 pass their likelihood ratios $\mathrm{LRReg}[\langle 1,\psi\rangle][0]$ for $0\le\psi\le 7$ to the blue nodes at layer 1 where new likelihood ratios $\mathrm{LRReg}[\langle 1,\psi\rangle][1]$ for $\psi=0,1,2,3$ are computed according to formula (13). Likewise, the newly computed likelihood ratios are passed forward and computed until node $L_8^{(1)}$ are finally reached and $\mathrm{LRReg}[\langle 1,0\rangle][3]$ is updated. All the nodes acitivated so far are blue nodes in Figure 3. At iteration $i=2$, node $L_8^{(2)}$ does not activate any node but computes $\mathrm{LRReg}[\langle 2,0\rangle][3]$ according to formula (14) given the values of $\mathrm{LRReg}[\langle 1,0\rangle][2]$ and $\mathrm{LRReg}[\langle 1,1\rangle][2]$ already calculated by two $L_4^{(1)}$ nodes as well as the value of $\mathrm{UReg}[\langle 1,0\rangle][3]$. Then node $L_8^{(2)}$ updates $\mathrm{UReg}[\langle 2,0\rangle][3]$ as in line 6 or 8 or 10 of Algorithm 1. Given $\mathrm{UReg}[\langle 1,0\rangle][3]$ and $\mathrm{UReg}[\langle 2,0\rangle][3]$, CalBit$(\cdot)$ calculates $\mathrm{UReg}[\langle 1,0\rangle][2]$ and $\mathrm{UReg}[\langle 1,1\rangle][2]$ using the 2-by-2 transform $G_2=\begin{bmatrix}1&1\\1&0\end{bmatrix}$. CalBit$(\cdot)$ is suspended for $i=2$. It cannot proceed to activate the four $L_2^{(1)}$ nodes to update the bit array UReg because $\mathrm{UReg}[\langle 2,0\rangle][2]$ and $\mathrm{UReg}[\langle 2,1\rangle][2]$ are yet to be available.

For every iteration i, CalLR$(\cdot)$ activates all the nodes in the same phase (i.e., the same color) and updates the corresponding elements in $\mathrm{LRReg}[N][m+1]$. CalBit$(\cdot)$ recursively calculates $\mathrm{UReg}[\langle\kappa,2\psi\rangle][m-1]$, $\mathrm{UReg}[\langle\kappa,2\psi+1\rangle][m-1]$ if both $\mathrm{UReg}[\langle 2\kappa-1,\psi\rangle][m]$ and $\mathrm{UReg}[\langle 2\kappa,\psi\rangle][m]$ are available. Every node in the array is activated once to update LRReg and one more time to update UReg leading to an overall computational complexity $O(N\log N)$.

[7] One may concern about the division arithmetic for safety reason. The recursive calculations of LRs can be replaced by recursive calculations of transition probabilities in Proposition 3 of [2] where only addition and multiplication are used.

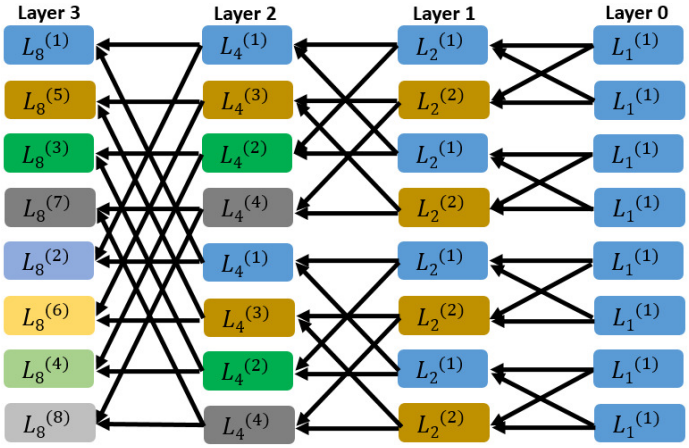

Fig. 3. The butterfly circuit.

2.4 Closeness Analysis of Polar Sampler

A good closeness metric can help reduce the complexity of implementation. In this section, we will evaluate the closeness error of the proposed sampling scheme.

Definition 2 (Kullback-Leibler divergence). *Let P and Q be two distributions over a common countable set Ω, and let $\mathcal{A} \subset \Omega$ be the strict support of P ($P(a) > 0$ iff $a \in \mathcal{A}$). The KL divergence D_{KL} of Q from P is defined as:*

$$D_{KL}(P\|Q) = \sum_{a\in\mathcal{A}} P(a) \ln\left(\frac{P(a)}{Q(a)}\right)$$

with the convention that $\ln(x/0) = +\infty$ for any $x > 0$.

Let $P_{X^{1:N}}(x^{1:N})$ denote the distribution of N i.i.d. Bernoullis X defined as above. For any $0 < \beta < 0.5$, $N = 2^n, n \geq 1$ and the corresponding high and low-entropy sets defined in (9) and (6), one can generate a distribution $Q_{X^{1:N}}(x^{1:N})$ using the rules (10) and (11). To give the KL divergence between $P_{X^{1:N}}(x^{1:N})$ and $Q_{X^{1:N}}(x^{1:N})$, we first modify the Bernoulli sampling rules (10) and (11) to be

$$U^{(i)} = \begin{cases} 0 \ w.p. \ \frac{1}{2} \\ 1 \ w.p. \ \frac{1}{2} \end{cases} \quad \text{if } i \in \mathcal{H}_{X|Y} \tag{15}$$

$$U^{(i)} = \begin{cases} 0 \ w.p. \ P_{U^{(i)}|Y^{1:N},U^{1:i-1}}(0|y^{1:N},u^{1:i-1}) \\ 1 \ w.p. \ P_{U^{(i)}|Y^{1:N},U^{1:i-1}}(1|y^{1:N},u^{1:i-1}) \end{cases} \quad \text{if } i \in \mathcal{H}^c_{X|Y}\ , \tag{16}$$

where only the deterministic decisions for $U^{(i)}$ in $\mathcal{L}_{X|Y}$ are replaced by random decisions. Let $Q'_{X^{1:N}}(x^{1:N})$ denote the distribution obtained from the new Bernoulli sampling rules described in (15) and (16).

Theorem 3 (Polar Sampling Theorem). *Let $X \sim P(X)$ be the target Bernoulli distribution with bias $c_1 = P(X = 1)$. To sample from the Bernoulli distribution $Ber(c_1)$, PolarSampler$(\cdot)$ in Algorithm 1 takes a bias vector $\boldsymbol{c} = [c_1, \cdots, c_1]$ of length N together with the high and low entropy set $\mathcal{H}_X, \mathcal{L}_X$ as input and computes N new Bernoulli biases $b_1, b_2, \cdots, b_N$ for a 2-power N. Sampling independent variables $U^{1:N}$ according to biases $b_1, b_2, \cdots, b_N$ and applying the transform G_N lead to a vector $X^{1:N} = U^{1:N}G_N$ of distribution $Q_{X^{1:N}}$. Let $Q'_{X^{1:N}}$ be the intermediate distribution defined earlier in this section. Then for $0 < \beta < 0.5$, we derive*

$$D_{KL}(P_{X^{1:N}} \| Q'_{X^{1:N}}) \leq 2 \ln 2 \cdot N 2^{-N^\beta} \text{ and } \quad D_{KL}(Q_{X^{1:N}} \| Q'_{X^{1:N}}) \leq \ln 2 \cdot N 2^{-N^\beta}.$$

In the presence of a side information $Y \in \mathcal{Y} \coloneqq \{y_1, \cdots, y_{|\mathcal{Y}|}\}$, X can be seen as a combination of a sequence of Bernoullis with biases $\boldsymbol{c} = [c_1, \cdots, c_{|\mathcal{Y}|}]$ for $c_p = P(X = 1|Y = y_p)$ and $y_p \in \mathcal{Y}$. PolarSampler$(\cdot)$ takes the bias vector $\boldsymbol{c}[y^{1:N}]$ and the high and low entropy set $\mathcal{H}_{X|Y}, \mathcal{L}_{X|Y}$ as input. The above closeness bound still holds by substituting $X|Y$ for X.

Proof. The KL divergence between $P_{X^{1:N}}$ and $Q'_{X^{1:N}}$ is bounded by the KL divergence between $P_{U^{1:N}|Y^{1:N}}$ and $Q'_{U^{1:N}|Y^{1:N}}$ because the following relation hold.

$$\begin{aligned}
&D_{KL}(P_{X^{1:N}} \| Q'_{X^{1:N}}) \leq D_{KL}(P_{X^{1:N},Y^{1:N}} \| Q'_{X^{1:N},Y^{1:N}}) \\
&\overset{(a)}{=} D_{KL}(P_{Y^{1:N}} \| Q'_{Y^{1:N}}) + D_{KL}(P_{X^{1:N}|Y^{1:N}} \| Q'_{X^{1:N}|Y^{1:N}}) \\
&\overset{(b)}{=} D_{KL}(P_{X^{1:N}|Y^{1:N}} \| Q'_{X^{1:N}|Y^{1:N}}) \overset{(c)}{=} D_{KL}(P_{U^{1:N}|Y^{1:N}} \| Q'_{U^{1:N}|Y^{1:N}}) \qquad (17)
\end{aligned}$$

The above equalities are derived as follows.
(a) The chain rule of KL divergence.
(b) $D_{KL}(P_{Y^{1:N}} \| Q'_{Y^{1:N}}) = 0$.
(c) One-to-one mapping between $U^{1:N}$ and $X^{1:N}$.

The conditional KL divergence $D_{KL}(P_{U^{1:N}|Y^{1:N}} \| Q'_{U^{1:N}|Y^{1:N}})$ is derived as

$$\begin{aligned}
&D_{KL}(P_{U^{1:N}|Y^{1:N}} \| Q'_{U^{1:N}|Y^{1:N}}) \\
&\overset{(d)}{=} \sum_{i=1}^{N} D_{KL}(P_{U^{(i)}|U^{1:i-1},Y^{1:N}} \| Q'_{U^{(i)}|U^{1:i-1},Y^{1:N}}) \\
&\overset{(e)}{=} \sum_{i \in \mathcal{H}_k} D_{KL}(P_{U^{(i)}|U^{1:i-1},Y^{1:N}} \| Q'_{U^{(i)}|U^{1:i-1},Y^{1:N}}) \\
&\overset{(f)}{=} \sum_{i \in \mathcal{H}_k} \ln 2 \left[1 - H_P(U^{(i)}|U^{1:i-1}, Y^{1:N})\right] \\
&\overset{(g)}{\leq} \sum_{i \in \mathcal{H}_k} \ln 2 \left[1 - Z_P(U^{(i)}|U^{1:i-1}, Y^{1:N})^2\right] \qquad (18) \\
&\overset{(h)}{\leq} 2 \ln 2 \cdot N 2^{-N^\beta}, \qquad (19)
\end{aligned}$$

where the equalities and inequalities are explained as follows.
(d) The chain rule of KL divergence.
(e) For $i \in \mathcal{H}^{\mathrm{c}}_{X|Y}$, $Q'(u^{(i)}|u^{1:i-1}, y^{1:N}) = P(u^{(i)}|u^{1:i-1}, y^{1:N})$.
(f) The definition of $D_{KL}(\cdot\|\cdot)$ and $Q'(U^{(i)}|u^{1:i-1}, y^{1:N}) = \frac{1}{2}$ for $i \in \mathcal{H}_{X|Y}$.
(g) $Z(X|Y)^2 \leq H(X|Y)$ [1].
(h) Definition of $\mathcal{H}_{X|Y}$ (9).

In a similar fashion, the KL divergence of $Q_{X^{1:N}_{1:r}}$ and $Q'_{X^{1:N}_{1:r}}$ is as follows.

$$
\begin{aligned}
&D_{KL}(Q_{X^{1:N}} \| Q'_{X^{1:N}}) \leq D_{KL}(Q_{U^{1:N}|Y^{1:N}} \| Q'_{U^{1:N}|Y^{1:N}}) \\
&= \sum_{i=1}^{N} D_{KL}(Q_{U^{(i)}|U^{1:i-1},Y^{1:N}} \| Q'_{U^{(i)}|U^{1:i-1},Y^{1:N}}) \\
&\overset{(i)}{=} \sum_{i \in \mathcal{L}_{X|Y}} D_{KL}(Q_{U^{(i)}|U^{1:i-1},Y^{1:N}} \| Q'_{U^{(i)}|U^{1:i-1},Y^{1:N}}) \\
&\overset{(j)}{=} \sum_{i \in \mathcal{L}_{X|Y}} \ln 2 \sum_{u^{1:i-1}, y^{1:N}} -Q'(u^{1:i-1}, y^{1:N}) \log Q'(\bar{u}^{(i)}|U^{1:i-1}, Y^{1:N}) \\
&\overset{(k)}{\leq} \sum_{i \in \mathcal{L}_{X|Y}} \ln 2 \cdot H_P(U^{(i)}|U^{1:i-1}, Y^{1:N}) \\
&\overset{(l)}{\leq} \sum_{i \in \mathcal{L}_{X|Y}} \ln 2 \cdot Z(U^{(i)}|U^{1:i-1}, Y^{1:N}) \qquad (20) \\
&\overset{(m)}{\leq} \ln 2 \cdot N 2^{-N^\beta}, \qquad (21)
\end{aligned}
$$

where the equalities and inequalities come from
(i) For $i \in \mathcal{L}^{c}_{X|Y}$, $Q'(u^{(i)}|u^{1:i-1}, y^{1:N}) = Q(u^{(i)}|u^{1:i-1}, y^{1:N})$.
(j) The definition of $D_{KL}(\cdot\|\cdot)$ (see Appendix A).
(k) See Appendix A.
(l) $H(X|Y) \leq Z(X|Y)$ [13].
(m) Corollary 1.

Therefore, the closeness measured by KL divergence can be concluded as

$$D_{KL}(P_{X^{1:N}} \| Q'_{X^{1:N}}) \leq 2 \ln 2 \cdot N 2^{-N^\beta} \text{and} \;\; D_{KL}(Q_{X^{1:N}} \| Q'_{X^{1:N}}) \leq \ln 2 \cdot N 2^{-N^\beta}.$$

Note that polar sampling without side information is easier. In the absence of Y, the above closeness analysis for KL divergence still hold by seeing Y independent of X.

Q.E.D.

Although we cannot give the KL divergence between $P_{X^{1:N}}$ and $Q_{X^{1:N}}$ due to the lack of triangle inequality, the absence of $D_{KL}(P_{X^{1:N}} \| Q_{X^{1:N}})$ will not prevent us from the security analysis which will be explained in Section 5.

3 Gaussian Sampling over the Integers Using Polar Sampler

Definition 3. *For any $c \in \mathbb{R}$, $s > 0$, define the discrete Gaussian distribution over $\mathbb{Z}$ as*

$$\forall x \in \mathbb{Z},\ D_{\mathbb{Z},c,s}(x) = \rho_{c,s}(x)/\rho_{c,s}(\mathbb{Z})$$

where $\rho_{c,s}(x) = \exp(-\pi|x-c|^2/s^2)$ and $\rho_{c,s}(\mathbb{Z}) = \sum_{z\in\mathbb{Z}} \rho_{c,s}(z)$.

In the above definition, the denominator $\rho_{c,s}(\mathbb{Z})$ is for normalization. For convenience, we may omit c for $c = 0$, e.g. $\rho_{0,s}(x) = \rho_s(x)$ and $D_{\mathbb{Z},0,s}(x) = D_{\mathbb{Z},s}(x)$.

Gaussian sampling over the integers $\mathbb{Z}$ can be formulated as a multilevel sampling problem over a binary partition chain $\mathbb{Z} \subset 2\mathbb{Z} \subset 4\mathbb{Z} \subset \cdots 2^r\mathbb{Z} \cdots$ of which each level is labeled by $X_1, X_2, \ldots, X_r, \ldots$ (see Fig. 4). Then the discrete Gaussian distribution over the integers $\mathbb{Z}$ induces a distribution $P_{X_{1:r}}$ whose limit is exactly $D_{\mathbb{Z},c,s}$ as r goes to infinity. By cutting off the tail area of negligible probability, a discrete Gaussian distribution over the integer lattice $\mathbb{Z}$ can be reduced to a distribution over a finite set. For example, if the cutoff points of $D_{\mathbb{Z},0,s=3\sqrt{2\pi}}$ are ± 16, the left and right tail areas are approximately 2^{-20}.

input : $\mathbf{c} = [\mathbf{c}_1, \cdots, \mathbf{c}_r]$,$\mathcal{H}_{X_k|X_{1:k-1}}$,$\mathcal{L}_{X_k|X_{1:k-1}}$ for $k = 1, \cdots, r$.
output: $x^{1:N}$

1 temp $=\mathbf{c}_1[1^{1:N}]$; // $\mathtt{c}_1 = P(X_1 = 1)$.
2 $x_1^{1:N}$=PolarSampler(temp,$\mathcal{H}_{X_1}$,$\mathcal{L}_{X_1}$)
3 temp=$\mathbf{c}_2[x_1^{1:N}]$; // $\mathtt{c}_2 = [P(X_2 = 1|X_1 = 0), P(X_2 = 1|X_1 = 1)]$.
4 $x_2^{1:N}$=PolarSampler(temp,$\mathcal{H}_{X_2|X_1}$,$\mathcal{L}_{X_2|X_1}$)
5 $\cdots$
6 temp=$\mathbf{c}_r[x_{1:r-1}^{1:N}]$;// $\mathtt{c}_r = [P(X_r = 1|X_{r-1}, \cdots, X_1 = 0\cdots 0), \cdots, P(X_r = 1|X_{r-1}, \cdots, X_1 = 1\cdots 1)]$
7 $x_r^{1:N}$=PolarSampler(temp,$\mathcal{H}_{X_r|X_{1:r-1}}$,$\mathcal{L}_{X_r|X_{1:r-1}}$)
8 **return** $x^{1:N} = x_1^{1:N} + 2 \cdot x_2^{1:N} + \cdots + 2^{r-1} \cdot x_r^{1:N}$

Algorithm 4: GaussianSampling($\cdot$).

We now begin to demonstrate how PolarSampler($\cdot$) can be used for discrete Gaussian sampling as in Algorithm 4 GaussianSampling($\cdot$). Suppose r levels of partition are employed to approximate $D_{\mathbb{Z},c,s}$. The chain rule of conditional probability and the chain rule of conditional entropy, i.e.

$$P(X_{1:r}) = \prod_{k=1}^{r} P(X_k|X_{1:k-1}) \text{ and } H(X_{1:r}) = \sum_{k=1}^{r} H(X_k|X_{1:k-1}),$$

imply that the Gaussian distribution over the finite constellation can be generated in a level-by-level way. For the k-th level, we can sample from the component

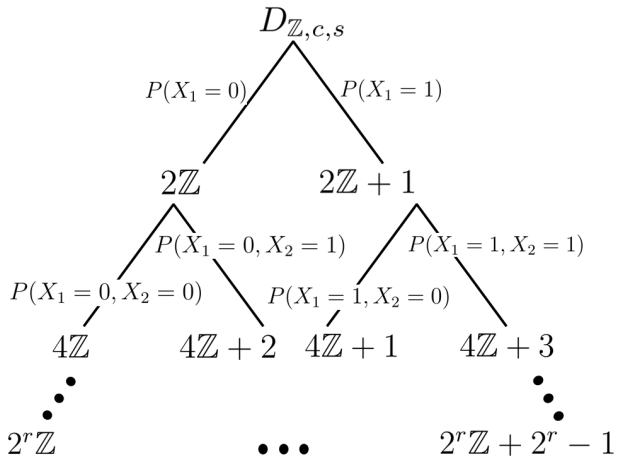

Fig. 4. An r-level binary partition tree of the integer lattice $\mathbb{Z}$.

source X_k using PolarSampler$(\cdot)$ given the samples $x_{1:k-1}$ from lower levels as side information. GaussianSampling$(\cdot)$ has access to a bias table $\mathbf{c} = [\mathbf{c}_1, \cdots, \mathbf{c}_r]$ defined as follows.

$$\begin{aligned}
&\mathbf{c}_1 = [P(X_1 = 1)]_{1\times 1}\\
&\mathbf{c}_2 = [P(X_2 = 1|X_1 = 0), P(X_2 = 1|X_1 = 1)]_{2\times 1}\\
&\quad \cdots\\
&\mathbf{c}_r = [P(X_r = 1|X_{1:r-1} = 0\cdots 0), \cdots, P(X_r = 1|X_{1:r-1} = 1\cdots 1)]_{2^{r-1}\times 1}
\end{aligned}$$

The high and low entropy sets $\mathcal{H}_{X_k|X_{1:k-1}}$ and $\mathcal{L}_{X_k|X_{1:k-1}}$ are computed according to the bias table $\mathbf{c}$ offline. We call this stage the construction stage of GaussianSampling$(\cdot)$. The online stage of GaussianSampling$(\cdot)$ draws Bernoulli samples level by level using PolarSampler$(\cdot)$ and we call it the implementation stage.

For the first level, we want to generate the component source X_1 in the absence of any side information.

1. *Construction:* By performing the source polarization transformation G_N on N i.i.d. copies of X_1, we obtain an N dimensional vector $U_1^{1:N} = X_1^{1:N}G_N$. For any $\beta \in (0, 1/2)$ and $\alpha = 2^{-N^\beta}$, we formally define two sets $\mathcal{H}_{X_1}$ and $\mathcal{L}_{X_1}$ as
$$\mathcal{H}_{X_1} = \left\{i \in [N] : \ Z(U_1^{(i)} \mid U_1^{1:i-1}) \in (1-\alpha, 1]\right\} \tag{22}$$
and
$$\mathcal{L}_{X_1} = \left\{i \in [N] : \ Z(U_1^{(i)} \mid U_1^{1:i-1}) \in [0, \alpha)\right\}. \tag{23}$$
For any $i \in \mathcal{H}_{X_1}$, $U_1^{(i)}$ is approximately uniform and independent of $U_1^{1:i-1}$, while for $i \in \mathcal{L}_{X_1}$, $U_1^{(i)}$ is almost deterministic given the knowledge of $U_1^{1:i-1}$.

2. *Implementation:* As discussed in Subsection 2.3, PolarSampler$(\cdot)$ calculates the posterior probability $P_{U_1^{(i)}|U_1^{1:i-1}}$ using SC decoding. Given the two sets $\mathcal{H}_{X_1}$, $\mathcal{L}_{X_1}$ and $P_{U_1^{(i)}|U_1^{1:i-1}}$, PolarSampler$(\cdot)$ generates N i.i.d. samples of X_1 by applying the polarization transform circuit to the vector $U_1^{1:N}$ of which each entry takes a value according to the following rule:

$$U_1^{(i)} = \begin{cases} \text{Bernoulli}(\frac{1}{2}) & \text{if } i \in \mathcal{H}_{X_1} \\ \arg\max_{u_1^{(i)}} P_{U_1^{(i)}|U_1^{1:i-1}}(u_1^{(i)}|u_1^{1:i-1}) & \text{if } i \in \mathcal{L}_{X_1} \end{cases}, \tag{24}$$

and

$$U_1^{(i)} = \begin{cases} 0 \ w.p. \ P_{U_1^{(i)}|U_1^{1:i-1}}(0|u_1^{1:i-1}) \\ 1 \ w.p. \ P_{U_1^{(i)}|U_1^{1:i-1}}(1|u_1^{1:i-1}) \end{cases} \text{if } i \in \mathcal{H}_{X_1}^c \backslash \mathcal{L}_{X_1} \ . \tag{25}$$

Once we have a realization $u_1^{1:N}$ of $U_1^{1:N}$, we derive a realization $x_1^{1:N} = u_1^{1:N} G_N$ of $X_1^{1:N}$ and pass it to the next level for further processing.

For higher levels with $k \in (1, r]$, our task is to generate N i.i.d. samples of source X_k given the side information $x_{1:k-1}^{1:N}$ which were generated at the previous $k-1$ levels.

1. *Construction:* By performing the source polarization transformation circuit G_N on N i.i.d. copies of X_k, we obtain an N dimensional vector $U_k^{1:N} = X_k^{1:N} G_N$. For $\beta \in (0, 1/2)$ and $\alpha = 2^{-N^\beta}$, we define $\mathcal{H}_{X_k|X_{1:k-1}}$ and $\mathcal{L}_{X_k|X_{1:k-1}}$ as

$$\mathcal{H}_{X_k|X_{1:k-1}} = \left\{ i \in [N] : \ Z(U_k^{(i)} \mid X_{1:k-1}^{1:N}, U_k^{1:i-1}) \in (1-\alpha, 1] \right\} \tag{26}$$

$$\mathcal{L}_{X_k|X_{1:k-1}} = \left\{ i \in [N] : \ Z(U_k^{(i)} \mid X_{1:k-1}^{1:N}, U_k^{1:i-1}) \in [0, \alpha) \right\}. \tag{27}$$

2. *Implementation:* Again PolarSampler$(\cdot)$ calculates $P_{U_k^{(i)}|X_{1:k-1}^{1:N}, U_k^{1:i-1}}$ using SC decoding. Then it generates N i.i.d. copies of X_k by applying the polarization transformation circuit to vector $U_k^{1:N}$ of which each entry takes a value according to the following rule:

$$U_k^{(i)} = \begin{cases} \text{Bernoulli}(\frac{1}{2}) & \text{if } i \in \mathcal{H}_{X_k|X_{1:k-1}} \\ \arg\max_u P_{U^{(i)}|X_{1:k-1}^{1:N}, U_k^{1:i-1}}(u|x_{1:k-1}^{1:N}, u_k^{1:i-1}) & \text{if } i \in \mathcal{L}_{X_k|X_{1:k-1}} \end{cases} \tag{28}$$

$$U_k^{(i)} = \begin{cases} 0 \ w.p. \ P_{U_k^{(i)}|X_{1:k-1}^{1:N}, U_k^{1:i-1}}(0|x_{1:k-1}^{1:N}, u_k^{1:i-1}) \\ 1 \ w.p. \ P_{U_k^{(i)}|X_{1:k-1}^{1:N}, U_k^{1:i-1}}(1|x_{1:k-1}^{1:N}, u_k^{1:i-1}) \end{cases} \\ \text{if } i \in \mathcal{H}_{X_k|X_{1:k-1}}^c \backslash \mathcal{L}_{X_k|X_{1:k-1}}. \tag{29}$$

Once we have a realization $u_k^{1:N}$ of $U_k^{1:N}$, we derive a realization $x_k^{1:N} = u_k^{1:N} G_N$ of $X_k^{1:N}$ and pass it to the next level for further processing. Recall that the approximation error for each level is determined by parameter β and

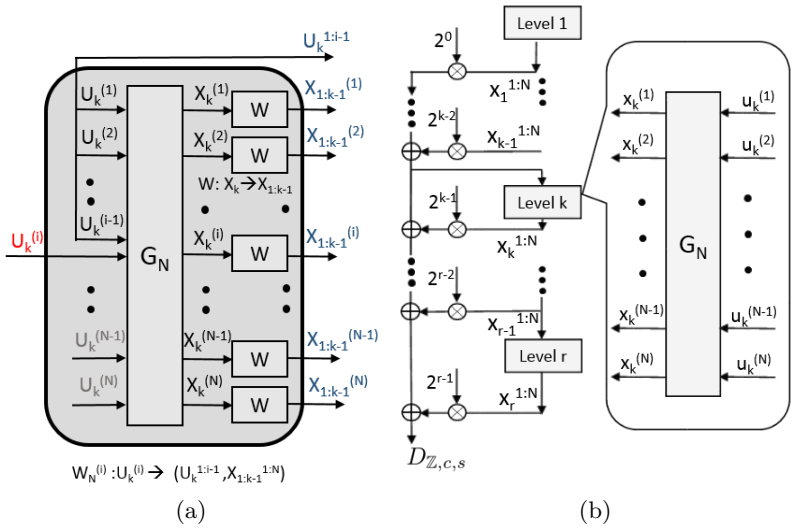

Fig. 5. The construction and implementation of the GaussianSampling(·): (a) Construction (can run offline) (b) Implementation (runs online).

N. To achieve a target closeness between the ideal distribution and the one we can produce, the two sets $\mathcal{H}_{X_k|X_{1:k-1}}$ and $\mathcal{L}_{X_k|X_{1:k-1}}$ for each level are properly chosen and determined offline. By repeating the operations in (28) and (29) from level 2 to level r, we can finally obtain N samples $x^{1:N}$ from $D_{\mathbb{Z},c,s}$, i.e.,

$$x^{1:N} = \sum_{k=1}^{r} 2^{k-1} x_k^{1:N}. \tag{30}$$

Fig. 5 shows how this Gaussian sampler works at each level in terms of construction and implementation. It also shows how to combine the output of each level. At the construction stage, W designates the probability transition from X_k to $X_{1:k-1}$ and $W_N^{(i)}$ designates the transition of synthesized source pair $(U_k^{(i)}, U_k^{1:i-1} \times X_{1:k-1}^{1:N})$. At this stage the Bhattacharyya parameters of $W_N^{(i)}$ are calculated to define $\mathcal{H}_{X_k|X_{1:k-1}}$ and $\mathcal{L}_{X_k|X_{1:k-1}}$. At the implementation stage, realizations of $U^{1:N}$ are produced according to the implementation rules (28) and (29). Given the two parameters N and β, the closeness between the ideal distribution and the one our sampler can produce will be analysed in next section.

4 Closeness Analysis of Discrete Gaussian Sampling

4.1 The Approximation Error Model

In a concrete implementation, an ideal discrete Gaussian distribution is replaced by an approximation. To give a sharp estimation of the accuracy/security of

a cryptographic primitive, the closeness between the ideal distribution and its approximation should be measured. In this section, we will derive the upper bounds on the closeness between the ideal distribution and the one generated by our sampling scheme measured by KL divergence.

The approximation error comes from two sources, the tailcut (finite levels of the partition employed) and the polar sampling. On the one hand, we need to decide how many levels of partition are needed. On the other hand, the error introduced by polar sampling should also be analysed. Denote by $D_{\mathbb{Z},c,s}$ the target discrete Gaussian distribution and we decide to employ r levels of partition. If polar sampling did not introduce any error, we would generate a distribution $P_{X_{1:r}}$ with a closeness measure $\delta(D_{\mathbb{Z},c,s}, P_{X_{1:r}})$ which is determined only by r for some metric δ. Then, let $Q_{X_{1:r}}(x_{1:r})$ denote the distribution obtained by polar sampling where the error introduced is $\delta(P_{X_{1:r}}, Q_{X_{1:r}})$. Since the KL divergence does not satisfy the triangle inequality, we will give $D_{KL}(D_{\mathbb{Z},c,s}\|P_{X_{1:r}})$ and $D_{KL}(P_{X_{1:r}}\|Q_{X_{1:r}})$ separately rather than a total KL divergence $D_{KL}(D_{\mathbb{Z},c,s}\|Q_{X_{1:r}})$. However, as discussed in [27, Chapter 3] the lack of symmetry and triangle inequality can be handled by a KL-based security argument which will be presented in Section 5.

4.2 Approximation Error from Tailcut

Definition 4 (Smoothing Parameter [21]). *For an n-dimensional lattice Λ, and positive real $\epsilon > 0$, we define its smoothing parameter $\eta_\epsilon(\Lambda)$ to be the smallest s such that $\rho_{1/s}(\Lambda^* \setminus \{0\}) \leq \epsilon$.*

The smoothing parameter quantifies how large s must be for $D_{\Lambda,c,s}$ to behave like a continuous Gaussian distribution. It is implied by Definition 4 that for any $\epsilon > 0$, the smoothing parameter $\eta_\epsilon(\mathbb{Z})$ of $\mathbb{Z}$ is the smallest s such that $\rho(s\mathbb{Z}) \leq 1 + \epsilon$.

Lemma 2 (Lemma 4.2, [12]). *For any $\epsilon > 0$, any $s > \eta_\epsilon(\mathbb{Z})$, and any $t > 0$,*

$$\Pr_{x \leftarrow D_{\mathbb{Z},c,s}}(|x - c| \geq t \cdot s) \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2e^{-\pi t^2}.$$

Instead of sampling over the full domain of the integer lattice, a distribution tail of negligible probability is cut off in practice. Suppose 2^r samples are left after the tailcut. Let $\mathcal{A} = \mathbb{Z} \cap [-2^{r-1} + c, 2^{r-1} + c)$. The distribution of the finite set is

$$D_\gamma(a) = \frac{\rho_{c,s}(a)}{\sum_{a \in \mathcal{A}} \rho_{c,s}(a)} = D_{\mathbb{Z},c,s}(a)/D_{\mathbb{Z},c,s}(\mathcal{A}),$$

where $a \in \mathcal{A}$ and γ is the probability of the tail. This constellation $\mathcal{A}$ of 2^r points can be represented as a binary partition tree labeled by $X_{1:r}$ in the same way as Fig. 4. In our sampling scheme, we obtain a sample labeled by

$$x^{1:N} = \sum_{k=1}^{r} 2^{k-1} x_k^{1:N}.$$

There exists a one-to-one mapping from $X_{1:r}$ to $\mathcal{A}$. Therefore $P_{X_{1:r}}$ and the tailcut distribution D_γ are exactly the same and we can obtain $D_{KL}(D_{\mathbb{Z},c,s}\|P_{X_{1:r}})$ by calculating $D_{KL}(D_{\mathbb{Z},c,s}\|D_\gamma)$. The distribution D_γ over the finite constellation $\mathcal{A}$ can be written in the form

$$P(X_{1:r}=x)=D_{\mathbb{Z},c,s}(a)/\sum_{a\in\mathcal{A}}D_{\mathbb{Z},c,s}(a)=D_{\mathbb{Z},c,s}(a|a\in\mathcal{A}).$$

The KL divergence between D_γ and $D_{\mathbb{Z},c,s}$ is

$$\begin{aligned}D_{KL}(D_\gamma\|D_{\mathbb{Z},c,s})&=\sum_{a\in\mathcal{A}}D_{\mathbb{Z},c,s}(a|a\in\mathcal{A})\ln\frac{D_{\mathbb{Z},c,s}(a|a\in\mathcal{A})}{D_{\mathbb{Z},c,s}(a)}\\&=\sum_{a\in\mathcal{A}}D_{\mathbb{Z},c,s}(a|a\in\mathcal{A})\ln\frac{D_{\mathbb{Z},c,s}(a|a\in\mathcal{A})}{D_{\mathbb{Z},c,s}(a|a\in\mathcal{A})D_{\mathbb{Z},c,s}(x\in\mathcal{A})}\\&=\sum_{a\in\mathcal{A}}D_{\mathbb{Z},c,s}(a|a\in\mathcal{A})\ln\frac{1}{D_{\mathbb{Z},c,s}(a\in\mathcal{A})}=\ln\frac{1}{D_{\mathbb{Z},c,s}(a\in\mathcal{A})}.\end{aligned}$$

According to the second-order Taylor bound, if $D_{\mathbb{Z},c,s}(a\in\mathcal{A})=1-\gamma$ for any $0<\gamma<1$, $D_{KL}(D_\gamma\|D_{\mathbb{Z},c,s})$ is bounded by

$$D_{KL}(D_\gamma\|D_{\mathbb{Z},c,s})=\gamma+O(\gamma^2). \tag{31}$$

and so is $D_{KL}(P_{X_{1:r}}\|D_{\mathbb{Z},c,s})$.

4.3 Approximation Error from Polar Sampling

The target discrete Gaussian distribution is tailcutted to be $P_{X_{1:r}}$ which is exactly the distribution of r bits of Bernoullis. Let $P_{X_{1:r}^{1:N}}$ denote the distribution of N i.i.d. $X_{1:r}$. As discussed in Section 3, for properly chosen $0<\beta<0.5$, $N=2^n, n\geq 1$ and the corresponding high and low-entropy sets defined in (26) and (27), one can approximate $P_{X_{1:r}^{1:N}}$ in a level-by-level manner using PolarSampler$(\cdot)$ for r times giving rise to the produced distribution $Q_{X_{1:r}^{1:N}}$.

Recall it in Theorem 3 that an intermediate distribution Q' is introduced to analyse the KL divergence betwewen P and Q. Likewise, for every $1\leq k\leq r$ we introduce an intermediate distribution $Q'_{X_k^{1:N}}$ such that $X_k^{1:N}=U_k^{1:N}G_N$ and

$$U_k^{(i)}=\begin{cases}0\ w.p.\ \frac{1}{2}\\1\ w.p.\ \frac{1}{2}\end{cases}\quad\text{if } i\in\mathcal{H}_{X_k|X_{1:k-1}}$$

$$U_k^{(i)}=\begin{cases}0\ w.p.\ P_{U_k^{(i)}|X_{1:k-1}^{1:N},U_k^{1:i-1}}(0|x_{1:k-1}^{1:N},u_k^{1:i-1})\\1\ w.p.\ P_{U_k^{(i)}|X_{1:k-1}^{1:N},U_k^{1:i-1}}(1|x_{1:k-1}^{1:N},u_k^{1:i-1})\end{cases}\quad\text{if } i\in\mathcal{H}^c{}_{X_k|X_{1:k-1}},$$

where only the deterministic decisions for $U_k^{(i)}$ in $\mathcal{L}_{X_k|X_{1:k-1}}$ are replaced by random decisions. We can bound the KL divergence between $P_{X_{1:r}^{1:N}}$ and $Q'_{X_{1:r}^{1:N}}$

as

$$
\begin{aligned}
&D_{KL}(P_{X_{1:r}^{1:N}} \| Q'_{X_{1:r}^{1:N}}) \overset{(a)}{=} D_{KL}(P_{U_{1:r}^{1:N}} \| Q'_{U_{1:r}^{1:N}}) \\
&\overset{(b)}{=} D_{KL}(P_{U_1^{1:N}} P_{U_2^{1:N}|U_1^{1:N}} \cdots P_{U_r^{1:N}|U_{1:r-1}^{1:N}} \| Q'_{U_1^{1:N}} Q'_{U_2^{1:N}|U_1^{1:N}} \cdots Q'_{U_r^{1:N}|U_{1:r-1}^{1:N}}) \\
&\overset{(c)}{=} \sum_{k=1}^{r} \sum_{i=1}^{N} D_{KL}(P_{U_k^{(i)}|U_k^{1:i-1},U_{1:k-1}^{1:N}} \| Q'_{U_k^{(i)}|U_k^{1:i-1},U_{1:k-1}^{1:N}}) \overset{(d)}{\leq} 2\ln 2 \cdot rN2^{-N^\beta}, \quad (32)
\end{aligned}
$$

where the equalities and inequalities are derived by (a) one-to-one mapping from $X_{1:r}^{1:N}$ to $U_{1:r}^{1:N}$; (b) the chain rule of joint distribution; (c) the chain rule of KL divergence; (d) Theorem 3. Likewise, we bound the KL divergence between $Q_{X_{1:r}^{1:N}}$ and $Q'_{X_{1:r}^{1:N}}$ as

$$
\begin{aligned}
&D_{KL}(Q_{X_{1:r}^{1:N}} \| Q'_{X_{1:r}^{1:N}}) = D_{KL}(Q_{U_{1:r}^{1:N}} \| Q'_{U_{1:r}^{1:N}}) \\
&= \sum_{k=1}^{r} \sum_{i=1}^{N} D_{KL}(Q_{U_k^{(i)}|U_k^{1:i-1},U_{1:k-1}^{1:N}} \| Q'_{U_k^{(i)}|U_k^{1:i-1},U_{1:k-1}^{1:N}}) \overset{(e)}{\leq} \ln 2 \cdot rN2^{-N^\beta}, \quad (33)
\end{aligned}
$$

where the inequality (e) is derived by Theorem 3. Although we cannot give the KL divergence between $P_{X_{1:r}^{1:N}}$ and $Q_{X_{1:r}^{1:N}}$ due to the lack of triangle inequality, the absence of $D_{KL}(P_{X_{1:r}^{1:N}} \| Q_{X_{1:r}^{1:N}})$ will not prevent us from the security analysis which will be explained in the sequel.

Remark 1. According to the KL-based analysis, GaussianSampling$(\cdot)$ can arbitrarily approximate $D_{\mathbb{Z}^{1:N},c,s}$ for sufficiently large N and properly chosen β and r. We highlight that the multilevel polar sampler would be attractive in applications consuming many more than one discrete Gaussian samples. There are plenty of appications of this kind in lattice-based cryptography and the prominent one is FHE. In FHE, it is quite common that dimension N can be tens of thousands. Even for lattice signature schemes (e.g. BLISS, Falcon), $N = 512,\ 1024$ is quite common, plus one may generate a batch of samples (except for embedded devices).

5 Security Analysis and Parameter Selection

5.1 Security Analysis with KL Divergence

Lemma 3 (Bounding Success Probability Variations, [26]). *Let $\mathcal{E}^{\mathcal{P}}$ be an algorithm making at most q queries to an oracle sampling from a distribution $\mathcal{P}$ and returning a bit. Let $\epsilon \geq 0$, and $\mathcal{Q}$ be a distribution such that $D_{KL}(\mathcal{P}\|\mathcal{Q}) < \epsilon$. Let x (resp. y) denote the probability that $\mathcal{E}^{\mathcal{P}}$ (resp. $\mathcal{E}^{\mathcal{Q}}$) outputs 1. Then, $|x - y| \leq \sqrt{q\epsilon/2}$.*

Security argument [27] It can be concluded from Lemma 3 that if a scheme is λ-bit secure with oracle access to a perfect distribution $\mathcal{P}$ and the KL divergence between $\mathcal{P}$ and another distribution $\mathcal{Q}$ satisfies $D_{KL}(\mathcal{P}\|\mathcal{Q}) \leq 2^{-\lambda}$, then this scheme is also about λ-bit secure with oracle access to $\mathcal{Q}$. Note that this security argument holds only if $\mathcal{E}$ is a search problem but not a decisional one. The security argument based on KL divergence satisfies symmetry and triangle inequality though KL divergence itself does not (see Section 3.2 in [27] for detail).

Consider that a scheme with access to a perfect distribution $D_{\mathbb{Z}^m,c,s}$ is λ-bit secure. Assume an adversary obtains m samples at each query for $m \leq N$. By the additivity of KL divergence and equation (31), we have $D_{KL}(D_{\mathbb{Z}^{1:m},c,s}\|P_{X_{1:r}^{1:m}}) \leq m(\gamma+\gamma^2)$. In order to achieve λ-bit security after the tailcut, we need to set $m(\gamma+\gamma^2) \approx 2^{-\lambda}$ by selecting $t \approx \sqrt{(\lambda+\log m)\ln 2/\pi}$. The number of levels needed is therefore $r = \lceil \log(2t \cdot s) \rceil$. As given in Section 4.3, the approximation error introduced by polar sampling is determined by both $D_{KL}(P_{X_{1:r}^{1:m}}\|Q'_{X_{1:r}^{1:m}})$ and $D_{KL}(Q'_{X_{1:r}^{1:m}}\|P_{X_{1:r}^{1:m}})$ which are upper bounded as

$$D_{KL}(P_{X_{1:r}^{1:m}}\|Q'_{X_{1:r}^{1:m}}) \leq 2\ln 2 \cdot rN2^{-N^\beta}, \quad D_{KL}(Q_{X_{1:r}^{1:m}}\|Q'_{X_{1:r}^{1:m}}) \leq \ln 2 \cdot rN2^{-N^\beta}.$$

In order to preserve λ-bit security after $P_{X_{1:r}^{1:m}}$ is replaced by $Q_{X_{1:r}^{1:m}}$, we need to select $n = \log N$ and β properly such that $2^{-2^{n\beta}+n+\log(r)+1} \approx 2^{-\lambda}$.

Fig. 6 shows how the security level of interest is related to β in terms of different s and n. Observe that the curves with the same n but different in s are quite close. It is understandable because the security is dependent on the approximation error as is almost independent of what the target distribution is if the proposed GaussianSampling($\cdot$) is used. At the implementation stage of GaussianSampling($\cdot$), the running time and entropy consumption to produce $U_k^{(i)}$ varies for polarized and unpolarized set. Smaller β implies smaller unpolarized set but larger approximation error. Parameter n is in a more dominant position because it greatly influences the computational complexity of SC decoding. Given optional choices of β and n to preserve λ-bit security, we suggest small β and small n for efficiency.

5.2 Security Analysis of Tailcut and Precision With Rényi Divergence

The KL-based security analysis is a reminder about Rényi divergence (RD). However, the approximation error of the proposed sampler measured by Rényi divergence is not given in this work because how polarization phenomenon converges in the metric of RD is an open problem in the area of coding theory. Nonetheless, RD can still be used to analyse the tailcut and precision.

Definition 5 (Rényi divergence). *Let P, Q be two distributions with supports $\mathcal{S}_P$ and $\mathcal{S}_Q$, respectively. Let $\mathcal{S}_P \subseteq \mathcal{S}_Q$. For $a \in (1, +\infty)$, we define the Rényi*

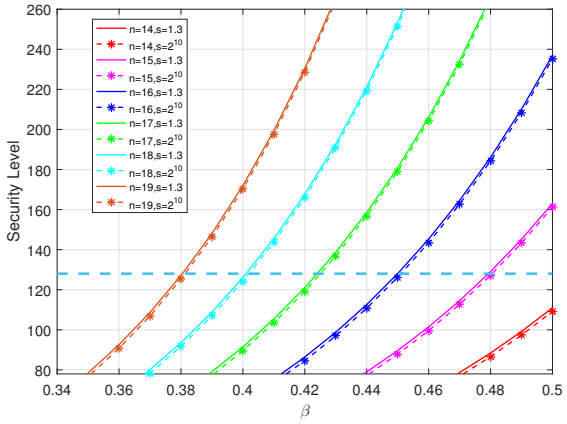

Fig. 6. Security level λ vs. β: $s = 1.3$ and $s = 2^{10}$.

divergence of order a by

$$R_a(P\|Q) = \left(\sum_{x\in\mathcal{S}_P} \frac{P(x)^a}{Q(x)^{a-1}}\right)^{\frac{1}{a-1}}.$$

In [28], a sharper bound on security based on Rényi divergence is given under the assumption that the number of adversary queries q to an λ-bit secure scheme is far less than 2^λ. Consider a cryptographic scheme with λ bits of security and the number of queries to $\mathcal{Q}$ verifies $q \le 2^{64}$. By the security argument in [28], this scheme is proved to lose at most one bit of security when $\mathcal{Q}$ is replaced by $\mathcal{Q}_\gamma$ provided that one of following conditions is satisfied.

(a) If $\mathcal{Q}_\gamma$ is the distribution under tailcut, then

$$\frac{\mathcal{Q}_\gamma}{Q} \le 1+\gamma \text{ for } \gamma = \frac{1}{4q}.$$

(b) If $\mathcal{Q}_\gamma$ denote a distribution having the same support with $\mathcal{Q}$ subject to some relative error, it should be satisfied that

$$1-\gamma \le \frac{\mathcal{Q}_\gamma}{Q} \le 1+\gamma \text{ for } \frac{\gamma^2}{(1-\gamma)^{a+1}} \le \frac{1}{4\lambda q}.$$

In the context of multilevel polar sampling, if $t \approx \sqrt{\frac{\ln 2(66+\log m)}{\pi}}$ and $r = \left\lceil \log\left(2s\sqrt{\frac{(66+\log m)\ln 2}{\pi}}\right)\right\rceil$, a $(\lambda+1)$-bit secure scheme will be at least λ-bit secure when the ideal distribution $D_{\mathbb{Z},c,s}$ is replaced by its tailcut $P_{X_{1:r}}$ for $\lambda = 128$, $q = 2^{64}$ and $a = 2\lambda$. Compared with the KL-based analysis of tailcut, Rényi divergence contributes to a reduction on r by at most 1 level.

Moreover, the security argument for relative error can be translated to $\gamma \leq 2^{-36.5}$ for $\lambda = 128$, $q = 2^{64}$ and $a = 2\lambda$. The precision requirement of polar sampler to achieve the target security is determined by the Bernoulli sampling in formula (28) and (29) which is yielded by SC decoding (i.e. CalLR$(\cdot)$ and CalBit$(\cdot)$). A practical approach to sample from Bernoulli is to calculate the bias Q_γ subject to the relative precision provided. Then sample $q \in (0, 1)$ uniformly at random and yield 1 if $q < Q_\gamma$. As long as the relative precision provided for SC decoding and the biases computed in formula (28) and (29) is more than 36 bits, polar sampler can achieve $\lambda = 128$. In our application, it suffices to use double precision in SC decoding which provides 52 bits of precision. It can also be simulated using fixed-point numbers of 64 bits of precision particularly in 64-bit architectures.

6 Complexity and Comparison

6.1 A Constant-Time Algorithm

Efforts have been made to develop constant-time solutions to existing samplers e.g. [14, 16] as well as proposing new ones endowed with this desired quality against timing attacks. At the implementation stage of GaussianSampling$(\cdot)$, the overall consumption of running time is dominated by two types of operations: calculating the LRs by SC decoding level by level and producing binary samples $U_{1:r}^{1:N}$ following the implementation rules as in formula (28) and (29). As distinguished from previous samplers, the GaussianSampling$(\cdot)$ relies on a recursive transform on N inputs and yields N samples at each run. For a fixed block length N, no matter what the distribution is, the SC decoding at each level requires exactly $N(1 + \log N)$ steps of LR calculations where two LRs are assembled to give a new one (formula (13) and (14)). Drawing binary samples according to (28) and (29) can also be made constant-time. On the one hand, drawing a bit in constant time uniformly at random or deterministically according to (28) is easy and cheap. On the other hand, how to sample Bernoulli variables $\mathcal{B}_p$ in (29) is given in Section 5.2 and can be converted to fixed-point operations. Moreover the fraction of unpolarized indexes vanishes as N tends to ∞. [8] In conclusion, the implementation stage shown in Fig. 5 is block-wise constant time if the two conditions below are satisfied:

- The number of levels r is fixed. Note that r is only related to the width s of $D_{\mathbb{Z},c,s}$ and the security level λ.
- The parameter β and N for each level is chosen properly such that $\mathcal{H}$, $\mathcal{L}$ and $\mathcal{H}^c \backslash \mathcal{L}$ for each level are fixed.

[8] The convergence speed scales with the block length as $N^{-\frac{1}{\mu}}$ where the upper bound on the scaling exponent μ depends solely on the specific distribution $D_{\mathbb{Z},c,s}$ and is bounded as $3.579 \leq \mu \leq 4.714$ (see Lemma 1).

6.2 Time Complexity

The latest trend of DGS solutions is to expand a base sampler into one for arbitrary parameters. For example, the Knuth-Yao and CDT sampler can work as a base sampler to produce samples which are then combined into new samples with a relatively large standard deviation in a convolutional manner [22, 26]. Our Gaussian sampler also admits such an extension for potential speedup and the focus of this subsection is to compare the GaussianSampling(·) with other base samplers. Karmakar et al. [16] compared the time complexity of Knuth-Yao and CDT showing that the former can be made more time-saving. Therefore, it is fair to compare our Gaussian sampler only with Knuth-Yao and we used a non constant-time Knuth-Yao implemented in C++[9] as well as its constant-time version[10]. We also perform benchmarks of a prototype GaussianSampling(·) in terms of different choices of parameter s in C++.

The experiment is conducted on a PC with Ubuntu 18.04 and an intel i9-9900K processor running at 3.60 GHz using one core. We use g++ to compile both Knuth-Yao and our implementation with compilation flag -Ofast enabled. For the benchmarks, we select $s \in \sqrt{2\pi} \cdot \{3, 8, 32, 256\}$ and a target security level $\lambda = 64$.[11] According to the KL-based security analysis in Section 5, we specify β to achieve 64 bits of security with respect to $N \in \{2^{13}, 2^{14}, 2^{15}\}$, and we select $r = \lceil \log(2st) \rceil$ where $t \approx \sqrt{(\lambda + \log m) \ln 2/\pi}$. We assume that the adversary obtains $m = 1024$ integer samples for each query to the sampling algorithm. The simulation results are shown in Table 1. Firstly, our Gaussian sampler always outperforms Knuth-Yao in speed with respect to the above setting. Secondly, Knuth-Yao slows down almost linearly as 2^r grows while GaussianSampling(·) still provides a competitive speed. This advantage stems from the binary partition of the integers. Thirdly, GaussianSampling(·) shows modest speed reduction as the block length increases from 2^{14} to 2^{15}. This doesn't contradict the asymptotic information optimality claim which implies less randomness consumption per sample for larger N. The polarization effect can reduce the consumption of randomness to the optimum and contributes to the speed. However, the overall running time, in the current implementation, is dominated by the block length. If N is not large enough such that the N source pairs are not polarized deeply enough, the acceleration gained from polarization may not be able to compensate for the slowdown induced by the increase of N.

6.3 Memory Cost

At the construction stage, we need to store a table $\mathbf{c}$ of biases, i.e., $P(x_k = 1 | x_{1:k-1})$ for $1 \leq k \leq r$. The table consists of $2^r - 1$ elements for the overall r levels. The biases are stored in natural order of $X_{1:k-1}$ such that once the samples for the

[9] https://github.com/AaronHall4/BKW-Algorithm
[10] https://github.com/jnortiz/HIBE-Gaussian-Sampling
[11] GaussianSampling(·) is extendable to other security levels (e.g. 128,256) in our double precision setting as discussed in Section 5.2.

Table 1. Comparison between the GaussianSampling(·) and Knuth-Yao.

2^r	s	Knuth-Yao (samples/second)		GaussianSampling(·) (samples/second)		
		constant-time	non constant-time	$N = 2^{13}$	$N = 2^{14}$	$N = 2^{15}$
2^6	$3\sqrt{2\pi}$	2.809E5/s	3.876E5/s	$\beta = 0.487$ 1.333E6/s	$\beta = 0.4535$ 1.283E6/s	$\beta = 0.4244$ 1.168E6/s
2^7	$8\sqrt{2\pi}$	1.172E5/s	2.212E5/s	$\beta = 0.4876$ 1.194E6/s	$\beta = 0.454$ 1.097E6/s	$\beta = 0.425$ 1.010E6/s
2^9	$32\sqrt{2\pi}$	3.255E4/s	6.017E4/s	$\beta = 0.488$ 0.960E6/s	$\beta = 0.4544$ 0.861E6/s	$\beta = 0.4252$ 0.792E6/s
2^{12}	$256\sqrt{2\pi}$	4.464E3/s	6.760E3/s	$\beta = 0.4885$ 0.680E6/s	$\beta = 0.455$ 0.621E6/s	$\beta = 0.4257$ 0.572E6/s

Table 2. Comparison of entropy, computational and storage complexity between multi-level polar sampler and existing samplers (λ: precision; $t*s$: tailcut; H/N: the fraction of high entropy set; $H(X)$: the entropy of X; μ is a constant bounded as $3.579 \leq \mu \leq 4.714$ according to Lemma 1).

	computational complexity/sample	entropy consumption/sample	storage
multilevel polar sampler	$O(t*s*\log(N))$ floating point[12]	$O(\lambda * N^{-\frac{1}{\mu}} + H/N) \to H(X)$	$O(\lambda * t * s)$
binary sampling	$O(\log(t*s))$ integer arithmetic	$> 6 + 3\log(s)$	$O(\lambda * \log(2.4 * t * s^2))$
CDT	$O(\log(t*s))$ binary search	λ	$O(\lambda * t * s)$
Knuth-Yao	$O(\log(t*s))$ Boolean functions	λ	$O(\log(t*s))$ Boolean functions

prior $k-1$ levels $X_{1:k-1}^{1:N}$ are ready we can find the corresponding bias by the index $x_{1:k-1}$ without scanning the table.

6.4 Asymptotic Comparison

We also give in Table 2 an asymptotic comparison of entropy consumption, computational and storage complexity between GaussianSampling(·) and other existing samplers, e.g. the binary sampling algorithm [9] and its constant-time version in [25, 11, 34], constant-time CDT [14], constant-time Knuth-Yao sampler[16].

The overall running time of GaussianSampling(·) depends on the SC decoding (LR computations) and Bernoulli sampling (entropy consumption). As indicated in Corollary 1, the fraction of unpolarized set scales as $N^{-\frac{1}{\mu}}$. When $N \to \infty$, the average entropy consumption approaches $H(X)$ which is the Shannon's entropy of the target distribution X.

[10] Or $O(t*s*\log\log N)$ floating point operations if the state-of-the-art SC decoding in [33] is used.

For binary sampling in BLISS [9], one sample requires entropy consumption of $O(6+3*\log(s))$. If we use a full-table access CDT for the base sampler and use a full-table Bernoulli sampler, the computational complexity would be $O(\log(t * s_0))$ (s_0: the parameter of base sampler) for binary search plus $O(\log(t * s))$ for integer arithmetic and the entropy cost will be much greater than $O(6+3*\log(s))$. Falcon uses bimodal Gaussian and rejection sampling which should have similar complexity to binary sampling.

The full-table access CDT [14] has a computational complexity of $O(\log(t * s))$ and requires a storage of $O(\lambda * t * s)$. For constant time Knuth-Yao [16], the computational complexity is $O(\log(t * s))$ Boolean function evaluations and the entropy consumption is $O(\lambda)$. It requires large program memory to store $O(\log(t * s))$ Boolean functions.

In conclusion, we claim our multilevel polar sampler to achieve the information-theoretic optimality and its prominent advantage is entropy consumption. For most sampling methods, the overall speed depends on computational complexity and randomness generation (e.g. producing Bernoulli samples in the binary sampling method). The computational complexity of our sampler is less attractive due to the $\log(N)$ factor but (a) multilevel polar sampler saves the time of producing randomness which is not considered as computational complexity but entropy consumption (b) our experiments in Table 1 show that multilevel polar sampler is much faster than Knuth-Yao. In addition, there is room to improve the computational efficiency seeing that the state-of-the-art SC decoding achieves a per-bit complexity of $O(\log \log N)$ [33].

7 Conclusions and Future Work

The polar sampler and its multilevel application for DGS is efficient, application-independent and constant-time. Our algorithm is effective in the case that a large number of samples from a certain distribution, e.g., integer Gaussian, are required. The optimization of entropy consumption stems from the polarization process in which the randomness moves to the high-entropy set. For fixed parameters, the construction stage is prepared offline and the implementation stage is carried out online and constant time. KL and Rényi divergence are used for security analysis, precision analysis and parameter selection. Since the Rényi divergence-based analysis of polar coding is still an open problem by now, it deserves more efforts to give a complete Rényi divergence-based analysis for our sampler and to carry out potential efficiency improvement.

In this paper, we only use the basic 2×2 kernel, whose finite-length performance is not the best. Optimizing finite-length performance using other kernels of polar codes as well as various other issues are left to future work.

References

1. Arıkan, E.: Source polarization. In: 2010 IEEE International Symposium on Information Theory. pp. 899–903 (June 2010)

2. Arıkan, E.: Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. IEEE Transactions on Information Theory **55**(7), 3051–3073 (2009)
3. Bai, S., Lepoint, T., Roux-Langlois, A., Sakzad, A., Stehlé, D., Steinfeld, R.: Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. Journal of Cryptology **31**(2), 610–640 (2018)
4. Barthe, G., Belaïd, S., Espitau, T., Fouque, P.A., Rossi, M., Tibouchi, M.: Galactics: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. p. 2147–2164. CCS '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3363223, https://doi.org/10.1145/3319535.3363223
5. Box, G.E.P., Muller, M.E.: A note on the generation of random normal deviates. Ann. Math. Statist. **29**(2), 610–611 (06 1958)
6. Bruinderink, L.G., Hülsing, A., Lange, T., Yarom, Y.: Flush, Gauss, and reload–a cache attack on the BLISS lattice-based signature scheme. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 323–345. Springer (2016)
7. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley & Sons (2012)
8. Devroye, L.: Sample-based non-uniform random variate generation. In: Proceedings of the 18th conference on Winter simulation. pp. 260–265. ACM (1986)
9. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Annual Cryptology Conference. pp. 40–56. Springer (2013)
10. Dwarakanath, N.C., Galbraith, S.D.: Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. Applicable Algebra in Engineering, Communication and Computing **25**(3), 159–180 (2014)
11. Espitau, T., Fouque, P.A., Gérard, B., Tibouchi, M.: Side-channel attacks on bliss lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. p. 1857–1874. CCS '17, Association for Computing Machinery, New York, NY, USA (2017)
12. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the fortieth annual ACM symposium on Theory of computing. pp. 197–206. ACM (2008)
13. Honda, J., Yamamoto, H.: Polar coding without alphabet extension for asymmetric models. IEEE Transactions on Information Theory **59**(12), 7829–7838 (2013)
14. Howe, J., Khalid, A., Rafferty, C., Regazzoni, F., O'Neill, M.: On practical discrete gaussian samplers for lattice-based cryptography. IEEE Transactions on Computers **67**(3), 322–334 (2016)
15. Hülsing, A., Lange, T., Smeets, K.: Rounded gaussians: fast and secure constant-time sampling for lattice-based crypto. In: Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Proceedings. pp. 728–757. Springer, Germany (2018)
16. Karmakar, A., Roy, S.S., Reparaz, O., Vercauteren, F., Verbauwhede, I.: Constant-time discrete Gaussian sampling. IEEE Transactions on Computers **67**(11), 1561–1571 (2018)
17. Knuth, D.: The complexity of nonuniform random number generation. Algorithm and Complexity, New Directions and Results pp. 357–428 (1976)

18. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1–23. Springer (2010)
19. Marsaglia, G., Tsang, W.W., et al.: The Ziggurat method for generating random variables. Journal of Statistical Software **5**(8), 1–7 (2000)
20. Micciancio, D., Peikert, C.: Hardness of SIS and LWE with small parameters. In: Advances in Cryptology–CRYPTO 2013, pp. 21–39. Springer (2013)
21. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. SIAM Journal on Computing **37**(1), 267–302 (2007)
22. Micciancio, D., Walter, M.: Gaussian sampling over the integers: Efficient, generic, constant-time. In: Annual International Cryptology Conference. pp. 455–485. Springer (2017)
23. Mondelli, M., Hashemi, S.A., Cioffi, J.M., Goldsmith, A.: Sublinear latency for simplified successive cancellation decoding of polar codes. IEEE Transactions on Wireless Communications **20**(1), 18–27 (2021)
24. Peikert, C.: An efficient and parallel Gaussian sampler for lattices. In: Annual Cryptology Conference. pp. 80–97. Springer (2010)
25. Pessl, P., Bruinderink, L.G., Yarom, Y.: To bliss-b or not to be: Attacking strongswan's implementation of post-quantum signatures. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. p. 1843–1855. CCS '17, Association for Computing Machinery, New York, NY, USA (2017)
26. Pöppelmann, T., Ducas, L., Güneysu, T.: Enhanced lattice-based signatures on reconfigurable hardware. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 353–370. Springer (2014)
27. Prest, T.: Gaussian sampling in lattice-based cryptography. Ph.D. thesis, École Normale Supérieure (2015)
28. Prest, T.: Sharper bounds in lattice-based cryptography using the Rényi divergence. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 347–374. Springer (2017)
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing. pp. 84–93. STOC '05, ACM, New York, NY, USA (2005)
30. Saarinen, M.J.O.: Arithmetic coding and blinding countermeasures for lattice signatures. Journal of Cryptographic Engineering **8**(1), 71–84 (Apr 2018)
31. Tal, I., Vardy, A.: List decoding of polar codes. IEEE Transactions on Information Theory **61**(5), 2213–2226 (2015)
32. Tal, I., Vardy, A.: How to construct polar codes. IEEE Transactions on Information Theory **59**(10), 6562–6582 (2013)
33. Wang, H.P., Duursma, I.M.: Log-logarithmic time pruned polar coding. IEEE Transactions on Information Theory **67**(3), 1509–1521 (2021)
34. Zhao, R.K., Steinfeld, R., Sakzad, A.: Facct: Fast, compact, and constant-time discrete gaussian sampler over integers. IEEE Transactions on Computers **69**(1), 126–137 (2019)

A KL Divergence for the Low-Entropy Set

For $i \in \mathcal{L}_{X_k|X_{1:k-1}}$, Q' and Q follow the distribution respectively as

$$Q'(u_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N}) = P(u_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N})$$
$$Q(\bar{u}_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N}) = 1 \text{ for } \bar{u}_k^{(i)} = \arg\max_{u\in\{0,1\}} P_{U_k^{(i)}|X_{1:k-1}^{1:N}, U_k^{1:i-1}}(u|x_{1:k-1}^{1:N}, u_k^{1:i-1}).$$

By definition of KL divergence, we have

$$\begin{aligned}
&D_{KL}(Q_{U_k^{(i)}|U_k^{1:i-1}, U_{1:k-1}^{1:N}} \| Q'_{U_k^{(i)}|U_k^{1:i-1}, U_{1:k-1}^{1:N}}) \\
&= \sum_{u_k^{1:i-1}, u_{1:k-1}^{1:N}} Q'(u_k^{1:i-1}, u_{1:k-1}^{1:N})[-1 \cdot \log Q'(\bar{u}_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N}) \\
&- 0 \cdot \log(1 - Q'(\bar{u}_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N})) + (0 \log 0 + 1 \log 1)].
\end{aligned}$$

By definition Shannon entropy,

$$\begin{aligned}
&H_P(U_k^{(i)}|U_k^{1:i-1}, U_{1:k-1}^{1:N}) \\
&= -\sum_{u_k^{1:i-1}, u_{1:k-1}^{1:N}} Q'(u_k^{1:i-1}, u_{1:k-1}^{1:N}) \sum_{u_k^{(i)}} Q'(u_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N}) \log Q'(u_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N})
\end{aligned}$$

for $i \in \mathcal{L}_{X_k|X_{1:k-1}}$. For $0.5 \le Q'(\bar{u}_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N}) < 1$ which is easily satisfied by choosing the low-entropy set $\mathcal{L}_{X_k|X_{1:k-1}}$ properly, we can prove that

$$\sum_{u_k^{(i)} \in \{\bar{u}_k^{(i)}, 1-\bar{u}_k^{(i)}\}} -Q'(u_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N}) \log Q'(u_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N}) \ge -\log Q'(\bar{u}_k^{(i)}|u_k^{1:i-1}, u_{1:k-1}^{1:N})$$

and $D_{KL}(Q_{U_k^{(i)}|U_k^{1:i-1}, U_{1:k-1}^{1:N}} \| Q'_{U_k^{(i)}|U_k^{1:i-1}, U_{1:k-1}^{N}}) \le H_P(U_k^{(i)}|U_k^{1:i-1}, U_{1:k-1}^{1:N})$.