

An MCMC Method to Sample from Lattice Distributions

Anand Jerry George

Department of Electrical Engineering
Indian Institute of Science, Bengaluru
Email: anandgeorge@iisc.ac.in

Navin Kashyap

Department of Electrical Communication Engineering
Indian Institute of Science, Bengaluru
Email: nkashyap@iisc.ac.in

Abstract—We introduce a Markov Chain Monte Carlo (MCMC) algorithm to generate samples from probability distributions supported on a d -dimensional lattice $\Lambda = \mathbf{B}\mathbb{Z}^d$, where \mathbf{B} is a full-rank matrix. Specifically, we consider lattice distributions P_Λ in which the probability at a lattice point is proportional to a given probability density function, f , evaluated at that point. To generate samples from P_Λ , it suffices to draw samples from a pull-back measure $P_{\mathbb{Z}^d}$ defined on the integer lattice. The probability of an integer lattice point under $P_{\mathbb{Z}^d}$ is proportional to the density function $\pi = |\det(\mathbf{B})|f \circ \mathbf{B}$. The algorithm we present in this paper for sampling from $P_{\mathbb{Z}^d}$ is based on the Metropolis-Hastings framework. In particular, we use π as the proposal distribution and calculate the Metropolis-Hastings acceptance ratio for a well-chosen target distribution. We can use any method, denoted by ALG, that ideally draws samples from the probability density π , to generate a proposed state. The target distribution is a piecewise sigmoidal distribution, chosen such that the coordinate-wise rounding of a sample drawn from the target distribution gives a sample from $P_{\mathbb{Z}^d}$. When ALG is ideal, we show that our algorithm is uniformly ergodic if $-\log(\pi)$ satisfies a gradient Lipschitz condition.

I. INTRODUCTION

Drawing samples from probability distributions is a ubiquitous problem in statistics and machine learning. In this paper, we look into the problem of sampling from probability distributions supported on a lattice (referred to as a *lattice distribution*). Specifically, we consider lattice distributions in which the probability at a lattice point is proportional to a given probability density function evaluated at that point. Such distributions have found applications in cryptography, lattice coding, and secure communication [1], [2], [3]. An example of a lattice distribution that has received much interest from researchers is the lattice Gaussian. A lattice Gaussian distribution is a probability distribution defined on a lattice Λ , such that for each $\mathbf{x} \in \Lambda$, the probability of \mathbf{x} is proportional to a Gaussian density function evaluated at \mathbf{x} . Lattice Gaussian sampling, also known as discrete Gaussian sampling (DGS), is closely related to shortest vector problem (SVP) and closest vector problem (CVP), which are computationally hard lattice problems [4]. Lattice Gaussian sampling is also employed for decoding and signal detection in Multiple Input Multiple Output (MIMO) systems [5]. There are some known methods

to generate samples from lattice Gaussian distributions [1], [6], [7], but few methods are available for drawing samples from arbitrary lattice distributions [8]. We try to address this problem. The motivation to go beyond lattice Gaussian is due to [3], where samples from a particular fat-tailed lattice distribution are used to achieve information-theoretically perfect security.

Since sampling methods for generic lattice distributions are not known prior to this work, we compare our algorithm with other available lattice Gaussian samplers. Algorithms currently available for sampling from d -dimensional lattice Gaussians require a sub-routine to generate samples from a 1-dimensional lattice Gaussian with arbitrary variance parameter. These schemes update each coordinate sequentially and therefore, cannot take advantage of the currently available optimized linear algebra libraries. Also, techniques used to sample from 1-dimensional lattice Gaussians are either computationally inefficient or require a pre-computed table [9]. This paper proposes a simple and provable algorithm that samples directly from d -dimensional lattice distributions, including lattice Gaussians.

Now we give a brief introduction to our algorithm. Let Λ be a d -dimensional lattice generated by a full-rank matrix \mathbf{B} . As mentioned earlier, we consider lattice distributions P_Λ in which the probability at a lattice point is proportional to the value of a given probability density function f at that point. By a simple reduction, it becomes evident that, to generate samples from P_Λ , it suffices to draw samples from a probability distribution $P_{\mathbb{Z}^d}$ defined on the integer lattice. The probability of an integer lattice point under $P_{\mathbb{Z}^d}$ is proportional to the probability density function

$$\pi(\mathbf{x}) = |\det(\mathbf{B})|f(\mathbf{B}\mathbf{x}) \quad \text{for all } \mathbf{x} \in \mathbb{R}^d. \quad (1)$$

This paper explores the possibility of using Markov chain Monte Carlo (MCMC) methods to draw samples from $P_{\mathbb{Z}^d}$. MCMC is a well-known paradigm in statistics to sample from the desired probability distribution by establishing a Markov chain whose stationary distribution is the same as the desired distribution. In particular, we use the well-known Metropolis-Hastings algorithm to establish a Markov chain with the desired stationary distribution (referred to as the *target distribution*). In a Metropolis-Hastings algorithm, a candidate for the next state of the Markov chain is drawn from a

The work of N. Kashyap was supported in part by a Swarnajayanti Fellowship awarded by the Dept. of Science and Technology (DST), Govt. of India.

particular *proposal distribution* and is then accepted as the next state with a certain *acceptance probability*. In contrast to the other instances in literature where Metropolis-Hastings algorithms are used for lattice Gaussian sampling [6], [7], we use a proposal distribution and target distribution that have probability density functions. In particular, we use π given in (1) as the proposal distribution. We can use any off-the-shelf method, denoted by **ALG**, that ideally draws samples from the probability density π , to generate a proposed state. In fact, this enables us to sample from lattice distributions beyond lattice Gaussians, i.e., when π is not a Gaussian density. We use a piecewise sigmoidal probability density as the target distribution, a sample from which, after coordinate-wise rounding, gives a sample from $P_{\mathbb{Z}^d}$. With this choice of the target distribution, we show that our algorithm has exponentially fast convergence (*uniform ergodicity*) if the lattice distribution is such that $-\log(\pi)$ satisfies a gradient Lipschitz condition. We also derive a bound on the rate at which our algorithm converges to the target distribution when **ALG** deviates from its ideality.

A. Organization of the Paper

The remainder of the paper is organized as follows. In Section II, we formalize the problem statement and recall some basics of MCMC and its convergence. In Section III, we describe our algorithm and its convergence analysis. Proposition 1 in Section III shows that our algorithm is uniformly ergodic if $-\log(\pi)$ satisfies a gradient Lipschitz condition. In Proposition 2, we derive a bound on the rate at which our algorithm converges to the target distribution when **ALG** is non-ideal. In Section IV, we present our algorithm's simulation results for three different target distributions: isotropic Gaussian distribution on \mathbb{Z}^d , isotropic Gaussian distribution on the Leech lattice, and "Perfect Security distribution" on \mathbb{Z}^d . Section V discusses the performance of our algorithm and compares it with Klein's algorithm [10], [1] for lattice Gaussian sampling.

II. PRELIMINARIES

A. Notations

We denote the state space of a Markov chain by \mathcal{X} . The operator \wedge operates on two numbers to output the minimum of them. Nearest integer point to a vector $\mathbf{x} \in \mathbb{R}^d$ obtained by coordinate-wise rounding is denoted by $\lfloor \mathbf{x} \rfloor$. We use $U[0, 1]$ to denote the uniform probability distribution on the interval $[0, 1]$. We denote the Borel-sigma algebra on \mathbb{R}^d by $\mathcal{B}(\mathbb{R}^d)$. For two probability measures μ and ν defined on the same probability space, we use the notation $\mu \ll \nu$ to indicate that μ is absolutely continuous with respect to ν .

B. Lattice Distributions

We now formally define a lattice and probability distributions defined on a lattice. Let $\mathbf{B} \in \mathbb{R}^{d \times d}$ be a full-rank matrix. The d -dimensional lattice Λ generated by \mathbf{B} is defined as

$$\Lambda := \{\mathbf{B}\mathbf{z} : \mathbf{z} \in \mathbb{Z}^d\}.$$

Any probability distribution defined with Λ as the support is known as a lattice distribution. In this paper, we look at a specific class of lattice distributions in which the probability distribution is induced by a density function on \mathbb{R}^d . That is, the probability of a lattice point is equal to the density function evaluated at that point with appropriate normalization. This paper mainly considers density functions having the following form

$$f(\mathbf{x}) = \frac{e^{-\psi(\mathbf{x})}}{Z_\psi} \quad \text{for all } \mathbf{x} \in \mathbb{R}^d,$$

where $\psi(\mathbf{x})$ is known as a *potential function*, and $Z_\psi = \int_{\mathbb{R}^d} e^{-\psi(\mathbf{x})} d\mathbf{x}$ is a normalization constant. However, in practice, the algorithm that we develop works for any lattice distribution induced by a density function. Let $P_\Lambda(\mathbf{x})$ for $\mathbf{x} \in \Lambda$ be a lattice distribution induced by the above $f(\mathbf{x})$, i.e.,

$$P_\Lambda(\mathbf{x}) = \frac{e^{-\psi(\mathbf{x})}}{Z} \quad \text{for all } \mathbf{x} \in \Lambda,$$

where

$$Z = \sum_{\mathbf{x} \in \Lambda} e^{-\psi(\mathbf{x})}.$$

Let \mathbf{B} denote a generator matrix of the lattice Λ . Then, for generating a sample from the probability distribution P_Λ , it suffices to sample from $P_{\mathbb{Z}^d}(\mathbf{z}) = P_\Lambda(\mathbf{B}\mathbf{z})$, where $\mathbf{z} \in \mathbb{Z}^d$, and then obtain \mathbf{x} as $\mathbf{x} = \mathbf{B}\mathbf{z}$. So, our problem reduces to one of sampling from the following probability distribution over \mathbb{Z}^d :

$$P_{\mathbb{Z}^d}(\mathbf{z}) = \frac{e^{-\psi(\mathbf{B}\mathbf{z})}}{Z} \quad \text{for all } \mathbf{z} \in \mathbb{Z}^d.$$

Let $\varphi(\mathbf{x}) := \psi(\mathbf{B}\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^d$, so that

$$P_{\mathbb{Z}^d}(\mathbf{z}) = \frac{e^{-\varphi(\mathbf{z})}}{Z} \quad \text{for all } \mathbf{z} \in \mathbb{Z}^d. \quad (2)$$

Also, let us define the probability density π as:

$$\pi(\mathbf{x}) := \frac{e^{-\varphi(\mathbf{x})}}{K} \quad \text{for all } \mathbf{x} \in \mathbb{R}^d, \quad (3)$$

where

$$K = \int_{\mathbb{R}^d} e^{-\varphi(\mathbf{x})} d\mathbf{x}.$$

Note that π is related to f as given in (1). We have now reduced the problem of sampling from a probability distribution defined on an arbitrary lattice Λ to sampling from a probability distribution defined on \mathbb{Z}^d . In the rest of this paper, we try to develop an algorithm for generating samples from the lattice distribution $P_{\mathbb{Z}^d}$ induced by the probability density π .

1) *Lattice Gaussian Distribution*: A lattice distribution that has received significant attention is the lattice Gaussian distribution. A lattice Gaussian distribution defined on a lattice Λ is given by

$$D_{\Lambda, \sigma, \mathbf{c}}(\mathbf{x}) = \frac{e^{-\frac{\|\mathbf{x} - \mathbf{c}\|^2}{2\sigma^2}}}{\sum_{\mathbf{y} \in \Lambda} e^{-\frac{\|\mathbf{y} - \mathbf{c}\|^2}{2\sigma^2}}} \quad \text{for all } \mathbf{x} \in \Lambda, \quad (4)$$

where $\mathbf{c} \in \mathbb{R}^d$ is the mean vector and σ is the variance parameter. Lattice Gaussian distributions have important practical applications, particularly in cryptography [1].

C. The Metropolis-Hastings Algorithm

As stated earlier, we take the MCMC route to generate samples from the desired lattice distribution. MCMC is a class of sampling algorithms in which a Markov chain is set up whose stationary distribution is the same as the desired probability distribution (also called the *target distribution*). The idea is to simulate this chain for a certain number of steps to draw samples approximately from the desired probability distribution.

Given a Markov chain, it is straightforward to find its stationary distribution. However, it is not apparent how to find a Markov chain with the desired stationary distribution. The Metropolis-Hastings algorithm provides a recipe for establishing a Markov chain with the desired stationary distribution. Let $\bar{\pi}$ denote the probability distribution from which we want to draw samples. Then, the Metropolis-Hastings algorithm consists of two steps in generating the next state of the Markov chain:

- Let \mathbf{x} be the current state. Generate a proposed state \mathbf{y} from some probability distribution $q(\mathbf{x}, \cdot)$ (referred to as the *proposal distribution*).
- Accept the proposed state as the next state of Markov chain with probability $\alpha(\mathbf{x}, \mathbf{y})$ given by

$$\alpha(\mathbf{x}, \mathbf{y}) = 1 \wedge \frac{\bar{\pi}(\mathbf{y})q(\mathbf{y}, \mathbf{x})}{\bar{\pi}(\mathbf{x})q(\mathbf{x}, \mathbf{y})}.$$

If the proposed state is independent of the current state, we call this the *Independent Metropolis-Hastings* algorithm. The acceptance ratio is then given by

$$\alpha(\mathbf{x}, \mathbf{y}) = 1 \wedge \frac{\bar{\pi}(\mathbf{y})q(\mathbf{x})}{\bar{\pi}(\mathbf{x})q(\mathbf{y})}.$$

From now on, we refer to the Markov chain associated with an Independent Metropolis-Hastings algorithm by MH Markov chain.

D. Distance between probability distributions

For assessing the goodness of any sampling algorithm, it is essential to have a metric defined on the space of probability distributions. The metric we use in our analysis is the Total Variation Distance (TVD). For two distributions μ and ν defined on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$, we use $\|\mu - \nu\|_{TV}$ to denote their TVD given by

$$\|\mu - \nu\|_{TV} = \sup_{A \in \mathcal{B}(\mathbb{R}^d)} |\mu(A) - \nu(A)|.$$

If λ is a probability measure such that $\mu \ll \lambda$ and $\nu \ll \lambda$, then an alternate expression for TVD is given by

$$\|\mu - \nu\|_{TV} = \frac{1}{2} \int_{\mathbb{R}^d} \left| \frac{d\mu}{d\lambda}(\mathbf{x}) - \frac{d\nu}{d\lambda}(\mathbf{x}) \right| \lambda(d\mathbf{x}), \quad (5)$$

where $\frac{d\mu}{d\lambda}$ and $\frac{d\nu}{d\lambda}$ are the Radon-Nikodym derivatives of μ and ν with respect to λ (see Lemma 2.1 in [11]).

E. Convergence to stationarity

In this section, we give some definitions useful in evaluating the convergence of a Markov chain to its stationary distribution. We refer the reader to [12], [13] for a comprehensive review of these topics.

Definition 1. A Markov chain with transition kernel P and stationary distribution $\bar{\pi}$ is *uniformly ergodic* if there exists $0 < \delta < 1$ and $M < \infty$ such that for all $\mathbf{x} \in \mathcal{X}$,

$$\|P^t(\mathbf{x}, \cdot) - \bar{\pi}(\cdot)\|_{TV} \leq M(1 - \delta)^t.$$

Theorem 1. (Theorem 8 in [13]). Let P be the transition kernel of a Markov chain and $\bar{\pi}$ be its stationary distribution. Suppose there exists a $\delta > 0$ and a probability measure ν such that, for all measurable $B \subseteq \mathcal{X}$,

$$P(\mathbf{x}, B) \geq \delta\nu(B) \quad \text{for all } \mathbf{x} \in \mathcal{X}.$$

Then the Markov chain with transition kernel P is uniformly ergodic and satisfies the following inequality:

$$\|P^t(\mathbf{x}, \cdot) - \bar{\pi}(\cdot)\|_{TV} \leq (1 - \delta)^t \quad \text{for all } \mathbf{x} \in \mathcal{X}.$$

Definition 2. A Markov chain with transition kernel P and stationary distribution $\bar{\pi}$ is *geometrically ergodic* if there exists $0 < \delta < 1$ such that, for all $\mathbf{x} \in \mathcal{X}$,

$$\|P^t(\mathbf{x}, \cdot) - \bar{\pi}(\cdot)\|_{TV} \leq M(\mathbf{x})(1 - \delta)^t,$$

with $M(\mathbf{x}) < \infty$.

Definition 3. For a small $\epsilon > 0$, and initial state \mathbf{x} , a *mixing time* $t_{\text{mix}}(\epsilon; \mathbf{x})$ is defined as

$$t_{\text{mix}}(\epsilon; \mathbf{x}) = \inf\{t : \|P^t(\mathbf{x}, \cdot) - \bar{\pi}(\cdot)\|_{TV} < \epsilon\}.$$

III. INDEPENDENT METROPOLIS-HASTINGS WITH ROUNDING (IMHR)

In this section, we introduce an Independent Metropolis-Hastings algorithm for sampling from lattice distribution $P_{\mathbb{Z}^d}$ defined in (2). In this algorithm, we suppose that it is possible to generate samples from the probability density π defined in (3). Any state-of-the-art MCMC algorithm such as Hamiltonian Monte Carlo (HMC) [14], or Metropolis adjusted Langevin algorithm (MALA) [15], can be used for this purpose. The idea is to use π as the proposal distribution in the Independent Metropolis-Hastings algorithm. For such a method to be effective in sampling from $P_{\mathbb{Z}^d}$, we need a target distribution $\bar{\pi}$ with the following properties:

- Using a random variable with probability distribution $\bar{\pi}$, we should be able to efficiently derive a random variable with distribution $P_{\mathbb{Z}^d}$.
- The probability distribution $\bar{\pi}$ should be statistically close to π . This will reduce the possibility of rejecting a proposal in the Independent Metropolis-Hastings algorithm, thereby improving its convergence speed to the stationary distribution.

A naive approach would be to choose $\bar{\pi}$ as a piecewise constant density. That is, $\bar{\pi}(\mathbf{x})$ is equal to $\pi(\lfloor \mathbf{x} \rfloor)$ with

appropriate normalization for all $\mathbf{x} \in \mathbb{R}^d$. It is easy to see that the rounding operation on a sample generated from $\bar{\pi}$ gives a sample from $P_{\mathbb{Z}^d}$. The drawback of such an approach is that the Markov chain thus generated need not be uniformly ergodic, even for lattice Gaussians (see Appendix A). This motivates us to find a $\bar{\pi}$ that is a better approximation to π .

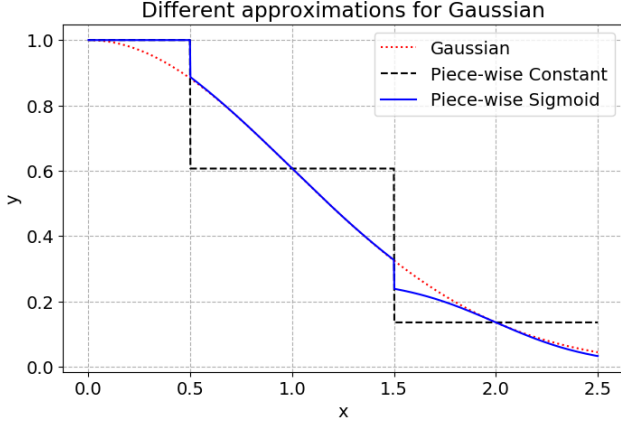


Fig. 1. Different approximations for Gaussian density

We define a new probability distribution $\bar{\pi}$ which will be called the *target distribution* henceforth, as follows:

$$\bar{\pi}(\mathbf{x}) = \frac{1}{Z} \frac{2e^{-\varphi(\bar{\mathbf{x}})}}{1 + e^{2(\mathbf{x}-\bar{\mathbf{x}})^T \nabla \varphi(\bar{\mathbf{x}})}} \quad \text{for all } \mathbf{x} \in \mathbb{R}^d, \quad (6)$$

where Z and φ are the same entities which appear in (2), and $\bar{\mathbf{x}}$ is the nearest integer point to \mathbf{x} which is obtained by coordinate-wise rounding. We should visualize $\bar{\pi}$ as a probability density function obtained by approximating π using a sigmoid function within each unit hypercube in \mathbb{R}^d and then normalizing. The sigmoid function is chosen such that, at the center of any unit hypercube, its value and gradient are proportional to the value and gradient of π . This is illustrated in Figure 1 where π is a Gaussian density function. Note that the functions plotted in Figure 1 are unnormalized. We can obtain a sample from $P_{\mathbb{Z}^d}$ by rounding the sample generated from $\bar{\pi}$ to its nearest integer point. To see this, let \mathbf{X} be a random variable with probability density $\bar{\pi}$. Let $\mathbf{Z} = \lfloor \mathbf{X} \rfloor$ and let S denote the unit hypercube centered at the origin, i.e., $S = [-\frac{1}{2}, \frac{1}{2}]^d$. Then,

$$\begin{aligned} \mathbb{P}(\mathbf{Z} = \mathbf{z}) &= \mathbb{P}(\mathbf{X} \in \mathbf{z} + S) \\ &= \int_S \bar{\pi}(\mathbf{z} + \mathbf{u}) d\mathbf{u} \\ &\stackrel{(a)}{=} \frac{1}{2} \int_S (\bar{\pi}(\mathbf{z} + \mathbf{u}) + \bar{\pi}(\mathbf{z} - \mathbf{u})) d\mathbf{u} \\ &= \frac{1}{Z} e^{-\varphi(\mathbf{z})} \int_S \left(\frac{1}{1 + e^{2\mathbf{u}^T \nabla \varphi(\mathbf{z})}} + \frac{1}{1 + e^{-2\mathbf{u}^T \nabla \varphi(\mathbf{z})}} \right) d\mathbf{u} \\ &= \frac{1}{Z} e^{-\varphi(\mathbf{z})} \int_S \left(\frac{1}{1 + e^{2\mathbf{u}^T \nabla \varphi(\mathbf{z})}} + \frac{e^{2\mathbf{u}^T \nabla \varphi(\mathbf{z})}}{1 + e^{2\mathbf{u}^T \nabla \varphi(\mathbf{z})}} \right) d\mathbf{u} \\ &= \frac{1}{Z} e^{-\varphi(\mathbf{z})}, \end{aligned}$$

where (a) is due to the symmetry of S . Therefore, to generate samples from $P_{\mathbb{Z}^d}$, it suffices to draw samples from $\bar{\pi}$ and then do coordinate-wise rounding.

Summarizing, IMHR is an Independent Metropolis-Hastings algorithm with π defined in (3) as the proposal distribution and $\bar{\pi}$ defined in (6) as the target distribution. The steps of IMHR are as described in Algorithm 1.

Algorithm 1: IMHR Algorithm

Input: $\mathbf{X}_0, \pi, \varphi, \nabla \varphi$

Output: Sample from a distribution statistically close to $P_{\mathbb{Z}^d}$

for $t = 1, 2, \dots$ **do**

 Let \mathbf{x} be the state of \mathbf{X}_{t-1} ;

 Generate \mathbf{y} from the probability distribution π ;

 Round \mathbf{x} to its nearest point in \mathbb{Z}^d to get $\bar{\mathbf{x}}$;

 Round \mathbf{y} to its nearest point in \mathbb{Z}^d to get $\bar{\mathbf{y}}$;

$$\bar{\pi}(\mathbf{x}) = \frac{2 \exp(-\varphi(\bar{\mathbf{x}}))}{1 + \exp(2(\mathbf{x}-\bar{\mathbf{x}})^T \nabla \varphi(\bar{\mathbf{x}}))};$$

$$\bar{\pi}(\mathbf{y}) = \frac{2 \exp(-\varphi(\bar{\mathbf{y}}))}{1 + \exp(2(\mathbf{y}-\bar{\mathbf{y}})^T \nabla \varphi(\bar{\mathbf{y}}))};$$

 Calculate acceptance ratio $\alpha(\mathbf{x}, \mathbf{y}) = 1 \wedge \frac{\bar{\pi}(\mathbf{y})\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})\pi(\mathbf{y})}$;

 Generate a sample u from $U[0, 1]$;

if $u \leq \alpha(\mathbf{x}, \mathbf{y})$ **then**

 let $\mathbf{X}_t = \mathbf{y}$;

else

$\mathbf{X}_t = \mathbf{x}$;

end

if $t > t_{\text{mix}}(\epsilon; \mathbf{X}_0)$ **then**

 Round \mathbf{X}_t to its nearest point in \mathbb{Z}^d to get $\bar{\mathbf{X}}_t$;

 Output $\bar{\mathbf{X}}_t$;

end

end

A. Convergence analysis of Algorithm 1

In this section, we analyze the convergence speed of Algorithm 1 to its stationary distribution. Algorithm 1 requires a sub-routine, denoted by **ALG** henceforth, which is ideally capable of drawing samples from π . The following analysis assumes that we have such a sub-routine available. Error due to non-availability of such an ideal sub-routine will be analyzed in the next section.

First, we state a well known theorem which is true in general for an Independent Metropolis-Hastings algorithm.

Theorem 2. (Theorem 2.1 in [16]) *An Independent Metropolis-Hastings algorithm is uniformly ergodic if there exist $\delta > 0$ such that*

$$\frac{\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})} \geq \delta \quad \text{for all } \mathbf{x} \in \mathcal{X}, \quad (7)$$

where π is the density from which proposed state is generated, and $\bar{\pi}$ is the target density. That is, the transition kernel \bar{P} of the MH Markov chain satisfies the following:

$$\|\bar{P}^k(\mathbf{x}, \cdot) - \bar{\pi}(\cdot)\|_{TV} \leq (1 - \delta)^k \quad \text{for all } \mathbf{x} \in \mathcal{X}.$$

Next, we define a widely used smoothness property of functions called L -smoothness.

Definition 4. A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is called L -smooth if the gradient of f is Lipschitz continuous with parameter L . That is, f should satisfy the following property

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

The following is a well known fact about L -smooth functions (see Lemma 5 in [17]). If f is L -smooth, then for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$,

$$f(\mathbf{y}) \leq f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^T \nabla f(\mathbf{x}) + \frac{L}{2} \|\mathbf{x} - \mathbf{y}\|^2. \quad (8)$$

Now we will give conditions on the probability density π that guarantees uniform ergodicity for Algorithm 1.

Proposition 1. Let π and $\bar{\pi}$ be as defined in (3) and (6) respectively. Let $\bar{\mathbf{x}}$ denote the nearest integer point to \mathbf{x} . If φ is an L -smooth function, then for all $\mathbf{x} \in \mathbb{R}^d$ we have,

$$\frac{\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})} = \frac{Z e^{-\varphi(\mathbf{x})} (1 + e^{2(\mathbf{x} - \bar{\mathbf{x}})^T \nabla \varphi(\bar{\mathbf{x}})})}{K 2e^{-\varphi(\bar{\mathbf{x}})}} \geq \frac{Z}{K} e^{-\frac{dL}{8}} > 0. \quad (9)$$

Therefore, by Theorem 2, Algorithm 1 is uniformly ergodic for such a π .

Proof: Let $\mathbf{y} = \mathbf{x} - \bar{\mathbf{x}}$. Then,

$$\begin{aligned} \frac{\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})} &= \frac{Z e^{\varphi(\bar{\mathbf{x}}) - \varphi(\bar{\mathbf{x}} + \mathbf{y})} (1 + e^{2\mathbf{y}^T \nabla \varphi(\bar{\mathbf{x}})})}{K 2} \\ &\stackrel{(a)}{\geq} \frac{Z e^{-\mathbf{y}^T \nabla \varphi(\bar{\mathbf{x}}) - \frac{L}{2} \|\mathbf{y}\|^2} (1 + e^{2\mathbf{y}^T \nabla \varphi(\bar{\mathbf{x}})})}{K 2} \\ &= \frac{Z e^{-\frac{L}{2} \|\mathbf{y}\|^2} (e^{\mathbf{y}^T \nabla \varphi(\bar{\mathbf{x}})} + e^{-\mathbf{y}^T \nabla \varphi(\bar{\mathbf{x}})})}{K 2} \\ &\stackrel{(b)}{\geq} \frac{Z}{K} e^{-\frac{L}{2} \|\mathbf{y}\|^2} \\ &\stackrel{(c)}{\geq} \frac{Z}{K} e^{-\frac{dL}{8}} > 0, \end{aligned}$$

where (a) follows from (8) due to L -smoothness of φ , (b) is due to AM-GM inequality and (c) is because $\mathbf{y} \in [-\frac{1}{2}, \frac{1}{2}]^d$. ■

Corollary 1. If π is a Gaussian density, then Algorithm 1 produces a uniformly ergodic Markov chain.

B. Effect of non-ideality of ALG

As defined in the previous section, **ALG** denotes the method used to draw samples from π in Algorithm 1. For the analysis in this section, we suppose that **ALG** is an MCMC method. In practice, **ALG** could be methods like HMC or MALA. By non-ideality of **ALG**, we mean that the TVD between the probability distribution from which **ALG** generate samples and π is nonzero. The non-ideality mentioned above can occur due to the finite time given for convergence in **ALG**. On account of this non-ideality, the proposed state in Algorithm 1 will have a probability distribution different from the one used in

the calculation of acceptance ratio (which is π). This alters the stationary distribution of Markov chain associated with Algorithm 1.

Suppose the Markov chain associated with **ALG** is geometrically ergodic for the stationary distribution π . In that case, we show in the following proposition that the error due to non-ideality of **ALG** can be bounded. For a discussion on the conditions under which methods like HMC and MALA are geometrically ergodic, we refer the reader to [18], [15].

Proposition 2. Let π defined in (3) be such that φ is L -smooth. Let $\mathbf{t} \in \mathbb{R}^d$ be a fixed initial state of **ALG**. Suppose **ALG** satisfies the following geometric ergodicity condition.

$$\|P^n(\mathbf{t}, \cdot) - \pi(\cdot)\|_{TV} \leq V\rho^n, \quad (10)$$

where P is the transition kernel corresponding to **ALG**. (V in the above expression may depend on the fixed initial state \mathbf{t} .) Also, let P be such that $\pi \ll P(\mathbf{t}, \cdot)$ and $P(\mathbf{t}, \{\mathbf{t}\}) > 0$. Then for any $\mathbf{x} \in \mathbb{R}^d$, Algorithm 1 generates a Markov chain with transition kernel \bar{P} that satisfies the following inequality:

$$\|\bar{P}^k(\mathbf{x}, \cdot) - \bar{\pi}(\cdot)\|_{TV} \leq (1 - C\delta)^k + \left(1 + \frac{1}{C\delta}\right) \frac{V\rho^n}{\delta}, \quad (11)$$

where $\bar{\pi}$ is the target distribution defined in (6), δ is obtained from (7), n is the number of iterations of **ALG**, k is the number of iterations of Algorithm 1, and C is a constant that satisfies the following inequality:

$$1 - \frac{2V\rho^n}{\delta} \leq C \leq 1 + \frac{2V\rho^n}{\delta}. \quad (12)$$

Proof: Let us denote $P^n(\mathbf{t}, \cdot)$ by $q(\cdot)$. By geometric ergodicity of **ALG** we have,

$$\|q - \pi\|_{TV} \leq V\rho^n. \quad (13)$$

Although **ALG** generates the proposed state from the distribution q , for calculating the acceptance ratio, we use the distribution π . Hence, the Markov chain generated by Algorithm 1 has the following transition kernel

$$\begin{aligned} \bar{P}(\mathbf{x}, d\mathbf{y}) &= q(d\mathbf{y}) \left(1 \wedge \frac{\bar{\pi}(\mathbf{y})\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})\pi(\mathbf{y})}\right) \\ &\quad + \delta_{\mathbf{x}}(d\mathbf{y}) \left(1 - \int_{\mathbb{R}^d} q(d\mathbf{z}) \left(1 \wedge \frac{\bar{\pi}(\mathbf{z})\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})\pi(\mathbf{z})}\right)\right), \end{aligned}$$

where $\delta_{\mathbf{x}}(\cdot)$ is the delta measure at \mathbf{x} . It is straightforward to verify using the detailed balance equation that the stationary distribution of the above Markov chain with transition kernel \bar{P} is given by

$$\nu(d\mathbf{x}) = \frac{\bar{\pi}(\mathbf{x})q(d\mathbf{x})}{C\pi(\mathbf{x})}, \quad (14)$$

where

$$C = \int_{\mathbb{R}^d} \frac{\bar{\pi}(\mathbf{x})q(d\mathbf{x})}{\pi(\mathbf{x})}. \quad (15)$$

Also, since φ is L -smooth, from Proposition 1, we have

$$\frac{\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})} \geq \delta > 0. \quad (16)$$

The first step in this proof is to show that the above Markov chain with transition kernel \bar{P} is uniformly ergodic. Then we show that its stationary distribution ν and probability density $\bar{\pi}$ are statistically close. These two results are combined to obtain the result stated in Proposition 2.

Uniform Ergodicity of \bar{P} :

From the expression for \bar{P} , for all $\mathbf{x} \in \mathbb{R}^d$ and all measurable $A \subseteq \mathbb{R}^d$, we have

$$\begin{aligned} \bar{P}(\mathbf{x}, A) &\geq \int_A q(d\mathbf{y}) \left(1 \wedge \frac{\bar{\pi}(\mathbf{y})\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})\pi(\mathbf{y})} \right) \\ &\stackrel{(a)}{=} \int_A C\nu(d\mathbf{y}) \frac{\pi(\mathbf{y})}{\bar{\pi}(\mathbf{y})} \left(1 \wedge \frac{\bar{\pi}(\mathbf{y})\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})\pi(\mathbf{y})} \right) \\ &= \int_A C\nu(d\mathbf{y}) \left(\frac{\pi(\mathbf{y})}{\bar{\pi}(\mathbf{y})} \wedge \frac{\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})} \right) \\ &\stackrel{(b)}{\geq} C\delta \int_A \nu(d\mathbf{y}) = C\delta\nu(A), \end{aligned}$$

where (a) and (b) are due to (14) and (16) respectively. Thus by Theorem 1, \bar{P} is the transition kernel of a uniformly ergodic Markov chain. Therefore,

$$\|\bar{P}^k(\mathbf{t}, \cdot) - \nu(\cdot)\|_{TV} \leq (1 - C\delta)^k. \quad (17)$$

Next, we find a bound on the value of C . Note that from the assumption $P(\mathbf{t}, \{\mathbf{t}\}) > 0$, it follows that for any measurable set $A \subseteq \mathbb{R}^d$ and integer n , $P^n(\mathbf{t}, A) > 0$ whenever $P(\mathbf{t}, A) > 0$. Therefore, $P(\mathbf{t}, \cdot) \ll q$. This, together with the assumption $\pi \ll P(\mathbf{t}, \cdot)$, allows us to conclude that $\pi \ll q$. Therefore, we have the following:

$$\begin{aligned} 1 &= \int_{\mathbb{R}^d} \bar{\pi}(d\mathbf{x}) \stackrel{(a)}{=} \int_{\mathbb{R}^d} \frac{\bar{\pi}(\mathbf{x})}{\pi(\mathbf{x})} \pi(d\mathbf{x}) \\ &\stackrel{(b)}{=} \int_{\mathbb{R}^d} \frac{\bar{\pi}(\mathbf{x})}{\pi(\mathbf{x})} \frac{d\pi}{dq}(\mathbf{x}) q(d\mathbf{x}), \end{aligned} \quad (18)$$

where (a) follows from the fact that π and $\bar{\pi}$ are density functions which are positive everywhere and (b) is due to the absolute continuity of π with respect to q . Then,

$$\begin{aligned} |C - 1| &\stackrel{(a)}{=} \left| \int_{\mathbb{R}^d} \frac{\bar{\pi}(\mathbf{x})}{\pi(\mathbf{x})} q(d\mathbf{x}) - \int_{\mathbb{R}^d} \frac{\bar{\pi}(\mathbf{x})}{\pi(\mathbf{x})} \frac{d\pi}{dq}(\mathbf{x}) q(d\mathbf{x}) \right| \\ &= \left| \int_{\mathbb{R}^d} \frac{\bar{\pi}(\mathbf{x})}{\pi(\mathbf{x})} \left(1 - \frac{d\pi}{dq}(\mathbf{x}) \right) q(d\mathbf{x}) \right| \\ &\leq \int_{\mathbb{R}^d} \frac{\bar{\pi}(\mathbf{x})}{\pi(\mathbf{x})} \left| 1 - \frac{d\pi}{dq}(\mathbf{x}) \right| q(d\mathbf{x}) \\ &\leq \frac{1}{\delta} \int_{\mathbb{R}^d} \left| 1 - \frac{d\pi}{dq}(\mathbf{x}) \right| q(d\mathbf{x}) \\ &\stackrel{(b)}{=} \frac{2}{\delta} \|q - \pi\|_{TV} \\ &\leq \frac{2V\rho^n}{\delta}, \end{aligned}$$

where (a) follows from (15) and (18), and (b) is due to the alternate definition of TVD given in (5). Therefore, we have

$$\begin{aligned} \delta - 2V\rho^n &\leq C\delta \leq \delta + 2V\rho^n \\ \implies 1 - \frac{2V\rho^n}{\delta} &\leq C \leq 1 + \frac{2V\rho^n}{\delta}. \end{aligned} \quad (19)$$

TVD between ν and $\bar{\pi}$:

Now we will show that ν and $\bar{\pi}$ are statistically close probability distributions.

$$\begin{aligned} \|\nu - \bar{\pi}\|_{TV} &\stackrel{(a)}{=} \frac{1}{2} \int_{\mathbb{R}^d} \left| \frac{d\nu}{dq}(\mathbf{x}) - \frac{d\bar{\pi}}{dq}(\mathbf{x}) \right| q(d\mathbf{x}) \\ &= \frac{1}{2} \int_{\mathbb{R}^d} \left| \frac{\bar{\pi}(\mathbf{x})}{C\pi(\mathbf{x})} - \frac{\bar{\pi}(\mathbf{x})}{\pi(\mathbf{x})} \frac{d\pi}{dq}(\mathbf{x}) \right| q(d\mathbf{x}) \\ &= \frac{1}{2C} \int_{\mathbb{R}^d} \frac{\bar{\pi}(\mathbf{x})}{\pi(\mathbf{x})} \left| 1 - C \frac{d\pi}{dq}(\mathbf{x}) \right| q(d\mathbf{x}) \\ &\leq \frac{1}{2C\delta} \int_{\mathbb{R}^d} \left| 1 - C \frac{d\pi}{dq}(\mathbf{x}) \right| q(d\mathbf{x}) \\ &\leq \frac{1}{2C\delta} \int_{\mathbb{R}^d} \left(C \left| 1 - \frac{d\pi}{dq}(\mathbf{x}) \right| + |C - 1| \right) q(d\mathbf{x}) \\ &= \frac{1}{\delta} \|q - \pi\|_{TV} + \frac{|C - 1|}{2C\delta} \\ &\stackrel{(b)}{\leq} \frac{V\rho^n}{\delta} + \frac{V\rho^n}{C\delta^2}, \end{aligned} \quad (20)$$

where (a) is due to the alternate definition of TVD given in (5), and (b) is due to (13) and (19).

Finally using triangle inequality, we have

$$\begin{aligned} \|\bar{P}^k(\mathbf{t}, \cdot) - \bar{\pi}(\cdot)\|_{TV} &\leq \|\bar{P}^k(\mathbf{t}, \cdot) - \nu(\cdot)\|_{TV} + \|\nu - \bar{\pi}\|_{TV} \\ &\leq (1 - C\delta)^k + (1 + \frac{1}{C\delta}) \frac{V\rho^n}{\delta}. \end{aligned} \quad (21)$$

IV. SIMULATION RESULTS

This section illustrates the speed of convergence of Algorithm 1 to $P_{\mathbb{Z}^d}$. For this, ideally we would like to show plots of TVD as a function of the number of iterations. However, evaluating distance between high dimensional probability distributions is computationally hard. So, in our simulations, we compute an entity TVD_m instead of TVD. We compute TVD_m as follows: Initialize Algorithm 1 with a fixed point in the state space. Then we run t iterations of Algorithm 1. Repeat this 100,000 times for each value of t . For each t , use these samples to form d 1-dimensional histograms, one for each coordinate. We call the distributions obtained by normalizing the histograms as the empirical marginal distributions, denoted by $h^i(\mathbf{z})$ for $1 \leq i \leq d$ and $\mathbf{z} \in \mathbb{Z}$. We denote the i^{th} marginal distribution of $P_{\mathbb{Z}^d}$ by $P_{\mathbb{Z}}^i$. If $P_{\mathbb{Z}}^i$ is not available in closed form, we estimate it using an MCMC method (see Appendix B-A for the exact algorithm used), with sufficient time given for

convergence. Calculate the Total Variation Distance between h^i and $P_{\mathbb{Z}}^i$ using the following formula:

$$\text{TVD}(i) = \frac{1}{2} \sum_{\mathbf{z} \in \mathbb{Z}} |h^i(\mathbf{z}) - P_{\mathbb{Z}}^i(\mathbf{z})| \quad \text{for } 1 \leq i \leq d.$$

Finally, TVD_m is the maximum of TVD's calculated for marginal distributions.

$$\text{TVD}_m = \max\{\text{TVD}(i) : 1 \leq i \leq d\}.$$

We plot TVD_m for different values of t . The TVD_m vs. t plots depicts the number of iterations required for Algorithm 1 to converge to its stationary distribution.

In another simulation, we plot the autocorrelation function of the time series $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N$ obtained using Algorithm 1. In many instances, autocorrelation plots have been used to assess the number of iterations of the Markov chain required to produce two almost independent samples [19]. The definition of the autocorrelation function that we use is as follows:

$$\text{ACF}(\tau) = \frac{\sum_{t=1}^{N-\tau} \mathbf{x}_t^T \mathbf{x}_{t+\tau}}{\sum_{t=1}^N \mathbf{x}_t^T \mathbf{x}_t},$$

where \mathbf{x}_t is the state of the Markov chain at iteration t and N is the total number of samples in the time series. Now we present results of these simulations for different probability densities π .

1) *Isotropic Gaussian distribution:* We first consider the case when π is an isotropic Gaussian density. The potential function φ of an isotropic Gaussian density is $\frac{1}{\sigma^2}$ -smooth, where σ^2 is the variance of the Gaussian. Therefore, from Proposition 1, we see that the factor that governs the rate of convergence of Algorithm 1 is $r = \frac{d}{\sigma^2}$. In this simulation, we fix $\sigma^2 = 1$ and vary dimension d to get different values of r . We use state $\mathbf{0}$ as the initial state of the algorithm. TVD_m vs. t for different values of r is shown in Figure 2. Autocorrelation vs. τ is shown in Figure 3. The number of samples (N) used to calculate the autocorrelation function is 10,000.

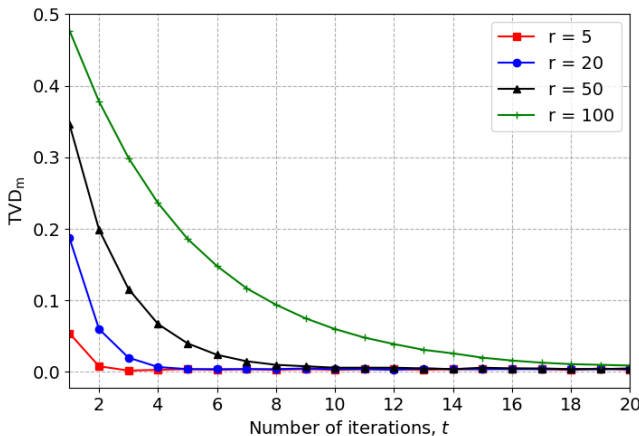


Fig. 2. TVD_m vs. Number of Iterations (t) for isotropic Gaussian

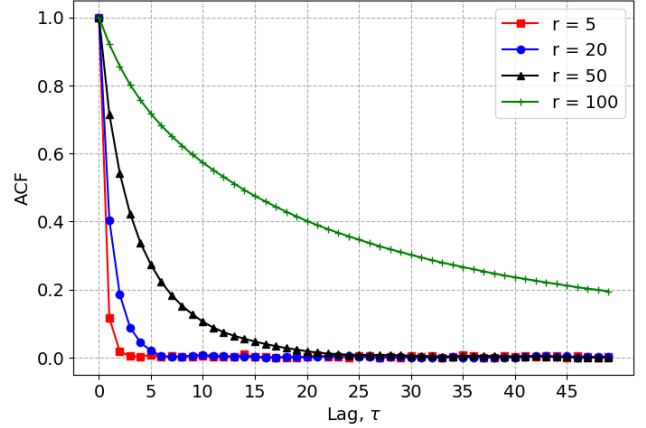


Fig. 3. ACF vs τ for isotropic Gaussian

2) *Gaussian distribution on the Leech lattice:* Next, we consider a lattice Gaussian distribution supported on the Leech lattice with dimension equal to 24 (see (23) for the generator matrix of the Leech lattice). This simulation illustrates the performance of Algorithm 1 for non-isotropic Gaussian. The Leech lattice induces a highly skewed lattice Gaussian distribution on \mathbb{Z}^d . The density π now takes the following form:

$$\pi(\mathbf{x}) = M e^{-\frac{\|\mathbf{B}\mathbf{x}\|^2}{2\sigma^2}} \quad \text{for all } \mathbf{x} \in \mathbb{R}^d,$$

where \mathbf{B} is the generator matrix of the Leech lattice and M is the normalization constant. We plot TVD_m vs. t for different values of σ^2 . This is shown in the Figure 4. State $\mathbf{0}$ was used as the initial state of the algorithm.

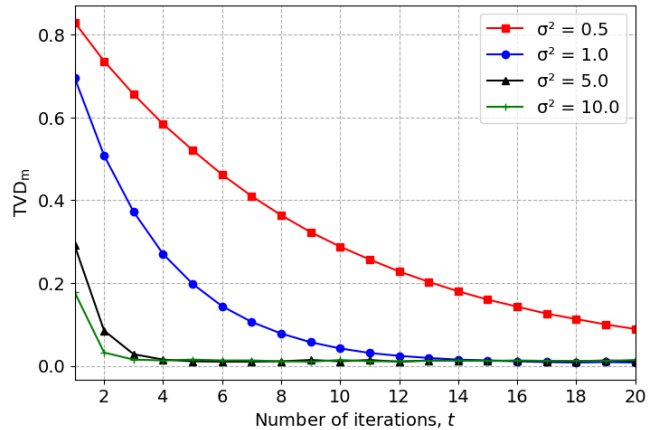


Fig. 4. TVD_m vs. Number of Iterations (t) for a Gaussian distribution on the Leech lattice

3) *Perfect Security distribution:* Finally, we present the simulation results when π is the following probability density which was used to achieve *perfect security* in [3]. The probability density function of a “Perfect Security distribution” is given by:

$$\pi(\mathbf{x}) = M \left(\frac{\Omega_d(\frac{\|\mathbf{x}\|}{2\rho})}{j_{\frac{d-2}{2}}^2 - \frac{\|\mathbf{x}\|^2}{4\rho^2}} \right)^2 \quad \text{for all } \mathbf{x} \in \mathbb{R}^d,$$

where

$$\Omega_d(u) = \left(\frac{2}{u}\right)^{\frac{d-2}{2}} J_{\frac{d-2}{2}}(u),$$

$J_k(u)$ is the Bessel function of k^{th} order and j_k is the first zero of k^{th} order Bessel function and M is the normalization constant. Let $\mathbf{X} = [X_1, X_2, \dots, X_d]$ be a random vector with perfect security probability distribution. Then, the variance of each component X_i is given by the following equation [20]:

$$\text{Var}(X_i) = \frac{4\rho^2 j_{\frac{d-2}{2}}^2}{d}.$$

We use HMC (see Appendix B-B for parameters used) to sample from this continuous density π . We plot TVD_m vs. t for different values of d . We fix the value of ρ such that the variance of the distribution is 1 for each value of d . The TVD_m vs. t plot is shown in Figure 5. State $\mathbf{0}$ was used as the initial state of the algorithm.

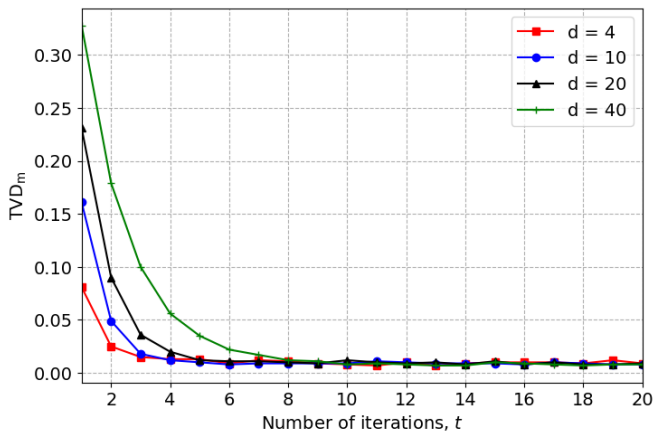


Fig. 5. TVD_m vs. Number of Iterations (t) for Perfect Security distribution

V. DISCUSSION

We presented a simple MCMC algorithm to draw samples from lattice distributions. As demonstrated through the Perfect Security distribution sampling, Algorithm 1 can sample from distributions beyond lattice Gaussians. To the best of our knowledge, prior to this work, there were no efficient algorithms known to generate samples from lattice distributions other than lattice Gaussians. The main feature of Algorithm 1, which makes it competitive even among the lattice Gaussian sampling algorithms, is its computational efficiency. The computations in Algorithm 1 are vector operations, which is highly optimized when current linear algebra libraries (for instance, OpenBLAS or Intel MKL) are used for implementation. Most of the algorithms currently available for sampling from a lattice Gaussian do coordinate-wise sequential sampling using 1-dimensional lattice Gaussian samplers. This method is inefficient when the lattice dimension under consideration is large.

A popular algorithm for sampling from lattice Gaussians is Klein's algorithm [10], [1]. In Figure 6, we compare the run-time per iteration of Klein's algorithm with Algorithm 1 for

different values of dimension when the desired distribution is a lattice Gaussian on \mathbb{Z}^d with variance parameter $\sigma = 1$. This experiment was run in a python environment on a machine with Intel i7-6700 @ 3.40GHz CPU and 8GB RAM. It is clear from Figure 6 that the scaling of the run-time per iteration with dimension is much better for Algorithm 1. However, multiple iterations of Algorithm 1 are required to generate a sample approximately from the stationary distribution. From Figure 2, we can obtain the minimum number of iterations of Algorithm 1 required to bring the TVD_m below a small number. Multiplying the run-time per iteration of Algorithm 1 by the minimum number of iterations, we see that the run-time required to generate a sample from lattice Gaussian is comparable for Klein's algorithm and Algorithm 1. For example, Algorithm 1 takes 13 iterations to bring the TVD_m below 0.005 when dimension equals 50. Run-time per iteration for Algorithm 1 is $46 \mu\text{s}$ when dimension equals 50. This implies a total run-time of $598 \mu\text{s}$ to generate a sample approximately from the stationary distribution. Klein's algorithm requires just one iteration to generate a sample from a lattice Gaussian distribution. However, it takes $798 \mu\text{s}$ per iteration.

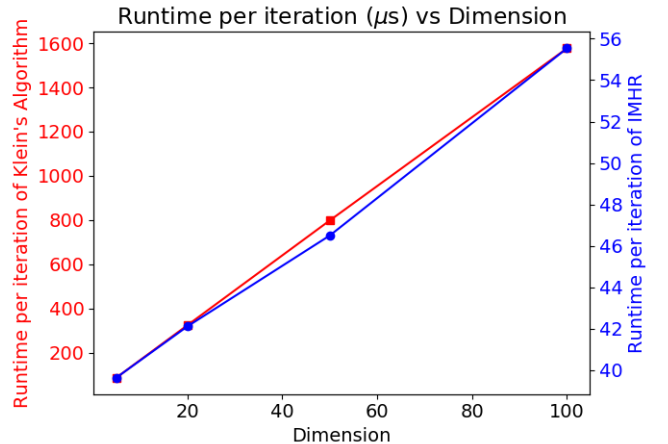


Fig. 6. Run-time vs Dimension for isotropic lattice Gaussian

From Proposition 1, we see that when π is an isotropic Gaussian density with variance equal to σ^2 , the TVD between the probability distribution after k^{th} iteration of Algorithm 1 and the stationary distribution is upper bounded by $(1 - \frac{Z}{K} e^{-\frac{d}{8\sigma^2}})^k$. This indicates that our algorithm may not be well suited for distributions with very low variance and high dimension. Figures 2 and 4 validate this by illustrating that convergence is slow for low variance and high dimension cases. In simulations, we observe that at high dimensions, the average acceptance, which is the fraction of iterations in which the proposed state is accepted, becomes very low for Algorithm 1. Figure 7 shows the degradation of average acceptance with dimension. A low acceptance ratio makes the Independent Metropolis-Hastings algorithm inefficient due to frequent rejection of the proposed state. Therefore, at very high dimensions, we suggest using the Metropolis-within-Gibbs strategy [21]. In the Metropolis-within-Gibbs algorithm, the

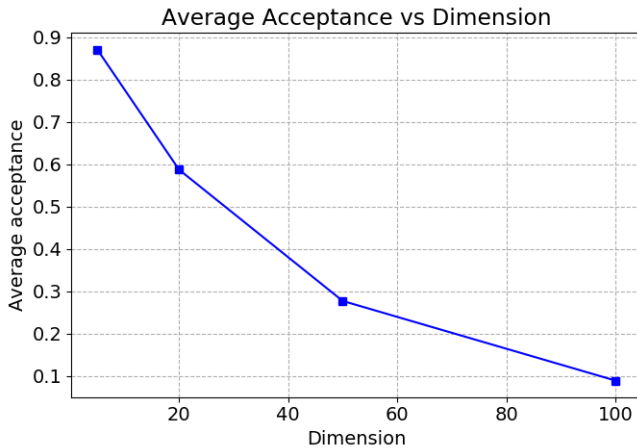


Fig. 7. Average Acceptance vs Dimension for isotropic lattice Gaussian

number of variables updated at a time, determines the average acceptance.

REFERENCES

- [1] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proc. 40th Annu. ACM Symp. Theory Comput.*, 2008, p. 197–206.
- [2] C. Ling and J. Belfiore, "Achieving AWGN channel capacity with lattice Gaussian coding," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5918–5929, 2014.
- [3] S. Vatedka, N. Kashyap, and A. Thangaraj, "Secure compute-and-forward in a bidirectional relay," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2531–2556, 2015.
- [4] D. Aggarwal, D. Dadush, O. Regev, and N. Stephens-Davidowitz, "Solving the Shortest Vector Problem in 2^n time using discrete Gaussian sampling: Extended abstract," in *Proc. STOC*, 2015, p. 733–742.
- [5] S. Liu, C. Ling, and D. Stehle, "Decoding by sampling: A randomized lattice algorithm for bounded distance decoding," *IEEE Trans. Inf. Theory*, vol. 57, no. 9, pp. 5933–5945, 2011.
- [6] Z. Wang and C. Ling, "On the geometric ergodicity of Metropolis-Hastings algorithms for lattice Gaussian sampling," *IEEE Trans. Inf. Theory*, vol. 64, no. 2, pp. 738–751, 2018.
- [7] Z. Wang, S. Lyu, and L. Liu, "Learnable Markov Chain Monte Carlo sampling methods for lattice Gaussian distribution," *IEEE Access*, vol. 7, pp. 87 494–87 503, 2019.
- [8] S. Anaswara, "Sampling from multidimensional distributions supported on a lattice," Master's thesis, Indian Institute of Science, Bengaluru, 2020.
- [9] J. Folláth, "Gaussian sampling in lattice based cryptography," *Tatra Mt. Math. Publ.*, vol. 60, pp. 1–23, 2014.
- [10] P. Klein, "Finding the closest lattice vector when it's unusually close," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2000, p. 937–941.
- [11] A. B. Tsybakov, *Introduction to Nonparametric Estimation*. Springer, 2008.
- [12] S. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*, 2nd ed. Cambridge University Press, 2009.
- [13] G. O. Roberts and J. S. Rosenthal, "General state space Markov chains and MCMC algorithms," *Probab. Surveys*, vol. 1, pp. 20–71, 2004.
- [14] R. M. Neal, "MCMC using Hamiltonian dynamics," in *Handbook of Markov chain Monte Carlo*, S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, Eds. Chapman and Hall/CRC, 2011, p. 113–162.
- [15] G. O. Roberts and R. L. Tweedie, "Exponential convergence of Langevin distributions and their discrete approximations," *Bernoulli*, vol. 2, no. 4, pp. 341–363, 1996.
- [16] K. L. Mengersen and R. L. Tweedie, "Rates of convergence of the Hastings and Metropolis algorithms," *Ann. Statist.*, vol. 24, no. 1, pp. 101–121, 1996.
- [17] R. Dwivedi, Y. Chen, M. J. Wainwright, and B. Yu, "Log-concave sampling: Metropolis-Hastings algorithms are fast," *J. Mach. Learn. Res.*, vol. 20, no. 183, pp. 1–42, 2019.
- [18] S. Livingstone, M. Betancourt, S. Byrne, and M. Girolami, "On the geometric ergodicity of Hamiltonian Monte Carlo," *Bernoulli*, vol. 25, no. 4A, pp. 3109–3138, 2019.
- [19] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng, Eds., *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 2011.
- [20] W. Ehm, T. Gneiting, and D. Richards, "Convolution roots of radial positive definite functions with compact support," *Trans. Am. Math. Soc.*, vol. 356, no. 11, pp. 4655–4685, 2004.
- [21] L. Tierney, "Markov chains for exploring posterior distributions," *Ann. Statist.*, vol. 22, no. 4, pp. 1701–1728, 1994.
- [22] S. F. Jarner and E. Hansen, "Geometric ergodicity of Metropolis algorithms," *Stoch. Process. Their Appl.*, vol. 85, no. 2, pp. 341–361, 2000.
- [23] G. O. Roberts, A. Gelman, and W. R. Gilks, "Weak convergence and optimal scaling of random walk Metropolis algorithms," *Ann. Appl. Probab.*, vol. 7, no. 1, pp. 110–120, 1997.

APPENDIX A
PIECE-WISE CONSTANT APPROXIMATION FOR GAUSSIAN
DENSITY

In this section, we substantiate the claim in Section III that the choice of $\bar{\pi}(\mathbf{x}) = \pi([\mathbf{x}])$ can give rise to a Markov chain which is not uniformly ergodic. We show this for a simple case where π is a 1-dimensional Gaussian density. From Theorem 2.1 in [16], it follows that, if $\text{ess inf } \frac{\pi(\mathbf{x})}{\bar{\pi}(\mathbf{x})} = 0$ with respect to $\bar{\pi}$ measure, then Independent Metropolis Hastings algorithm is not even geometrically ergodic. Let π be a 1-dimensional Gaussian density and $\bar{\pi}(x) = \pi([x])$. Let \bar{x} denote $[x]$ and let $y = x - \bar{x}$. Then,

$$\begin{aligned} \frac{\pi(x)}{\bar{\pi}(x)} &= M \frac{e^{-(\bar{x}+y)^2}}{e^{-\bar{x}^2}} \\ &= M e^{-2\bar{x}y} e^{-y^2}, \end{aligned} \quad (22)$$

where M is a constant. By definition, essential infimum of $\frac{\pi(x)}{\bar{\pi}(x)}$ with respect to $\bar{\pi}$ measure is the greatest number a such that the set,

$$A = \{x \in \mathbb{R} : \frac{\pi(x)}{\bar{\pi}(x)} < a\}$$

has zero $\bar{\pi}$ -measure. It is clear from (22) that, by choosing a large value for x , $\frac{\pi(x)}{\bar{\pi}(x)}$ can be made arbitrary close to 0 within a set of nonzero $\bar{\pi}$ measure.

$$\implies \text{ess inf } \frac{\pi(x)}{\bar{\pi}(x)} = 0.$$

This shows that for $\bar{\pi}(\mathbf{x}) = \pi([\mathbf{x}])$, Independent Metropolis Hastings algorithm need not even be geometrically ergodic.

APPENDIX B
OTHER MCMC METHODS USED

A. MCMC method used in the estimation of marginal distributions

This section elaborates on the MCMC method used for estimating marginal distributions $P_{\mathbb{Z}}^i$ required to calculate TVD_m in Section IV. We use the Random Walk Metropolis (RWM) algorithm to estimate the marginal distributions. This is a Metropolis-Hastings algorithm in which proposal density $q(\mathbf{x}, \mathbf{y})$ is a function of $\|\mathbf{y} - \mathbf{x}\|$. We refer an interested reader to [22], [23] for more on random walk Metropolis algorithms. Target distribution $\bar{\pi}$ used in this algorithm is the following piece-wise constant density derived from π .

$$\bar{\pi}(\mathbf{x}) = \pi([\mathbf{x}]) \quad \text{for all } \mathbf{x} \in \mathbb{R}^d.$$

Due to the symmetric nature of the proposal density, the acceptance ratio takes the following simple form:

$$\alpha(\mathbf{x}, \mathbf{y}) = 1 \wedge \frac{\bar{\pi}(\mathbf{y})}{\bar{\pi}(\mathbf{x})}.$$

In particular, we use the random walk Metropolis algorithm with proposal density $q(\mathbf{x}, \cdot)$ being a Gaussian density with mean \mathbf{x} and covariance matrix Σ . We choose Σ to be proportional to the covariance matrix of π .

Algorithm 2 describes the steps involved in the random walk Metropolis. We do 500 iterations of this algorithm to give it enough time to converge to the stationary distribution and thereby generate one sample. We use 200,000 such samples to form the histogram for each co-ordinate, which gives us the estimate of marginal distributions.

Algorithm 2: Random Walk Metropolis Algorithm

Input: $\pi, \Sigma, \mathbf{X}_0$

Output: Sample from a distribution statistically close to $P_{\mathbb{Z}^d}$

for $t = 1, 2, \dots$ **do**

Let \mathbf{x} denote the state of \mathbf{X}_{t-1} ;

Generate \mathbf{w} from $\mathcal{N}(0, \Sigma)$;

$\mathbf{y} \leftarrow \mathbf{x} + \mathbf{w}$;

Round \mathbf{y} to its nearest point in \mathbb{Z}^d to get $\bar{\mathbf{y}}$;

Round \mathbf{x} to its nearest point in \mathbb{Z}^d to get $\bar{\mathbf{x}}$;

Calculate acceptance ratio $\alpha(\mathbf{x}, \mathbf{y}) = 1 \wedge \frac{\pi(\bar{\mathbf{y}})}{\pi(\bar{\mathbf{x}})}$;

Generate a sample u from $U[0, 1]$;

if $u \leq \alpha(\mathbf{x}, \mathbf{y})$ **then**

let $\mathbf{X}_t = \mathbf{y}$;

else

$\mathbf{X}_t = \mathbf{x}$;

end

if $t > t_{\text{mix}}(\epsilon; \mathbf{X}_0)$ **then**

Round \mathbf{X}_t to its nearest point in \mathbb{Z}^d to get $\bar{\mathbf{X}}_t$;

end

end

B. Hamiltonian Monte Carlo (HMC)

HMC is used to generate samples from perfect security distribution in section IV-3. We refer the reader to [14] for an exposition on HMC. The input parameters to HMC are the number of Leapfrog steps (L) and the Leapfrog step-size (ϵ). The values of L and ϵ used in our simulation are as given below:

$$\begin{aligned} L &= \left\lceil 5 \left(\frac{2}{d} \right)^{\frac{1}{4}} \right\rceil, \\ \epsilon &= 1.2 \left(\frac{2}{d} \right)^{\frac{1}{4}}. \end{aligned}$$

Inside HMC, we resample momentum variables from an isotropic Gaussian density with variance equal to 9. We use five iterations of HMC to approximately generate a sample from the perfect security distribution.

