Latent RAGE: Randomness Assessment using Generative Entropy Models

Kuheli Pratihar, *Student Member, IEEE*, Rajat Subhra Chakraborty *Senior Member, IEEE,* and Debdeep Mukhopadhyay, *Senior Member, IEEE*

***Abstract*—NIST's recent review of the widely employed Special Publication (SP) 800-22 randomness testing suite has underscored several shortcomings, particularly the absence of entropy source modeling and the necessity for large sequence lengths. Motivated by this revelation, we explore low-dimensional modeling of the entropy source in random number generators (RNGs) using a Variational Autoencoder (VAE). This low-dimensional modeling enables the separation between strong and weak entropy sources by magnifying the deterministic effects in the latter, which are otherwise difficult to detect with conventional testing. Bits from weak-entropy RNGs with bias, correlation, or deterministic patterns are more likely to lie on a low-dimensional manifold within a high-dimensional space, in contrast to strong-entropy RNGs such as true random number generators (TRNGs) and pseudo-random number generators (PRNGs) with uniformly distributed bits. We exploit this insight to employ a generative AI-based Non-Interference Test (GeNI) for the first time, achieving implementation-agnostic low-dimensional modeling of all types of entropy sources. GeNI's generative aspect uses VAEs to produce synthetic bitstreams from the latent representation of RNGs, which are subjected to a Deep Learning (DL) based non-interference (NI) test evaluating the masking ability of the synthetic bitstreams. The core principle of the NI test is that if the bitstream exhibits high-quality randomness, the masked data from the two sources should be indistinguishable. GeNI facilitates a comparative analysis of low-dimensional entropy source representations across various RNGs, adeptly identifying the artificial randomness in specious RNGs with deterministic patterns that otherwise passes all NIST SP800-22 tests. Notably, GeNI achieves this with $10\times$ lower sequence lengths and $16.5\times$ faster execution time compared to the NIST test suite.**

***Index Terms*—Randomness, NIST, GenAI, Autoencoder, VAE, Non-interference, Deep Learning.**

I. Introduction

RANDOM number generators (RNGs) are integral components critical to cryptographic systems. They are essential in generating keys and masks and influence the security strength of protocols like Media Access Control Security (MACsec) and Transport Layer Security (TLS) [1]. Cryptographically secure RNGs are typically classified into two categories: true random number generators (TRNGs) and pseudo-random number generators (PRNGs). The former extracts randomness from physical processes, ensuring non-deterministic noise, while the latter extends the seed into a longer sequence with good statistical properties. The quality of RNGs is of paramount importance for security, as weak or predictable sequences may compromise the security of cryptographic systems. One notable instance occurred in 2012, when an analysis of millions of RSA public keys revealed that 0.2% of them could be factored using Euclid's algorithm due to shared prime [2]. This vulnerability arose from poorly seeded PRNGs, resulting in the creation of weak keys. Additionally, in December 2010, the group fail0verflow recovered the Elliptic Curve Digital Signature Algorithm (ECDSA) private key used by Sony to sign software for the PlayStation 3 [3]. This breach was possible because Sony failed to generate a new random nonce for each signature, allowing attackers to compromise the system's security. These cases underscore the critical importance of robust RNGs in safeguarding cryptographic systems. To uphold their integrity, RNGs undergo rigorous testing during design, production, and in-field operation, utilizing standard statistical test suites (STS).

In this context, NIST Special Publication (SP) 800-22 is a popularly employed randomness testing suite [4], which provides a standardized testing framework with 15 recommended tests for assessing the randomness of sequences. However, recent public comments [5] on the NIST proposal to revise the SP 800-22 randomness testing suite have raised some concerning comments listed as follows:

- "Either add stronger wording at the start of the document, rejecting it's use for assessing secure RNGs or withdraw the document entirely" from David Johnston [5].
- "The SP 800-22 document mainly finds use by amateur cryptographers and vendors of insecure systems, as it trivializes random bit generator validation to black-box statistical testing. Its use as evidence of security usually signals that competent specialists have not been involved in the design and analysis of a random bit generator." from Saarinen et al. [6].

These specific comments are just the tip of the iceberg as detailed in the public comments. The two most pressing concerns highlighted frequently in [5] are described the below:

1) **Stochastic modeling of entropy sources** is not sufficiently addressed in the NIST SP 800-22, which focuses on the statistical properties of sequences, ignoring the physical properties of the entropy source [6]. The lack of stochastic modeling leads to a) the statistical testing approach not relating to computational indistinguishably [6], [7], i.e.,

Manuscript received XX August 2024; accepted XX August 2024. Date of publication YY October 2024; date of current version ZZ October 2024. This article will be presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS) 2024 and appears as part of the ESWEEK-TCAD special issue. This work was supported in part by Qualcomm (Reference #IND-492686). (*Corresponding author: Kuheli Pratihar*)

Kuheli Pratihar, Rajat Subhra Chakraborty, and Debdeep Mukhopadhyay are with the Department of Computer Science and Engineering, Indian Institue of Technology, Kharagpur, India.

how hard it is to "break" the random and pseudorandom generators is not considered, and b) promoting false confidence in insecure systems as the weakest pseudorandom number generators will easily pass these tests, for example, compressible strings with specious randomness [8] as shown in Fig. 1.

2) **Long sequence length requirement** for most NIST tests demands prolonged data collection periods and increases the complexity of statistical analysis, leading to higher computational overheads and potentially slower detection of anomalies. They further lead to increased complexity for practical on-the-fly testing [9], exacerbating the vulnerabilities to certain types of attacks, such as those targeting specific segments of the sequence or exploiting temporal variations in the data generation process.

In response to the above shortcomings, a review was initiated by NIST's Crypto Publication Review Board, which decided to revise SP 800-22 Rev.1a [10]. The key revision objectives included clarifying the STS's purpose and discouraging its use for assessing cryptographic RNGs. Motivated by this revelation, in this work, we introduce a Generative Artificial Intelligence-based Non-Interference (GeNI) test designed to address a crucial limitation of the NIST SP 800-22 test suite: the lack of entropy source modeling. To the best of our knowledge, we, for the first time, leverage GenAI to model entropy sources in RNGs. GeNI leverages Variational Autoencoders (VAEs), which are popularly used for synthetic image and text generation [11]–[13], and more recently in diffusion transformers for text-to-image generation [14]. We repurposed the VAE to assess the quality of the entropy source directly from the bitstream by making suitable modifications to the fundamental architecture. Our approach differs from its conventional application in image generation in several key aspects including the following:

1) **Encoder-Decoder Network Architecture:** Instead of using convolutional neural networks (CNNs) typically employed for feature extraction in images, we utilize multilayer perceptrons (MLPs) to learn the latent entropy parameters in the bitstreams, processing binary values through the encoder-decoder network, unlike the multi-bit pixel representation in traditional VAEs for images.
2) **Low-Dimensional Latent Space:** VAE with a low latent dimension can effectively learn features such as texture, shape, and other characteristics, following the manifold hypothesis, which posits that high-dimensional data points, especially those from real-world sources, typically reside on or near a low-dimensional manifold within the high-dimensional space [15]. We employ this insight for RNGs, by choosing a low latent space dimension for VAEs to distinguish between strong and weak entropy sources.
3) **Evaluating quality of VAE output:** To extend the methodology to test TRNGs we needed to come out with a novel *non-interference* based test, which utilizes the masking capability of the synthetic bit streams to detect the non-interference property. We employ MLPs to classify between two masked bitstreams, thereby assessing the quality of randomness in the mask and, consequently, the input bitstream of the VAE.

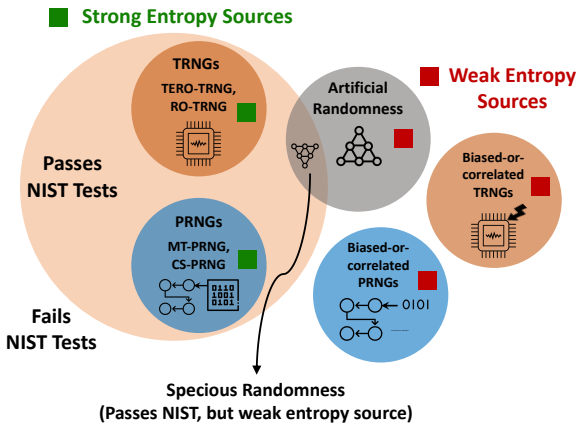

Fig. 1. Landscape of RNGs and NIST SP800-22 tests.

In summary, this is the first reported approach to adapt the VAEs to model the entropy source in RNGs. Hence, the above changes, namely choosing the underlying architecture, interpreting the goodness, and designing the goodness test, were introduced to cater VAEs to RNG tests. Moreover, the adaptability of VAEs, enables GeNI to accommodate various entropy source characteristics, rendering it suitable for a diverse range of RNG implementations.

Subsequently, we demonstrate that the proposed non-interference (NI) test employed on the synthetic bitstream output of the VAEs is a necessary and sufficient test to evaluate the quality of randomness of RNGs. We first establish that, formally, the masked data from the sources will be indistinguishable if and only if the mask is random. Thereafter, we employ a deep neural network (DNN) to evaluate the synthetic bitstreams' ability to mask the two distinct distributions. Drawing from principles of the non-interference theory [16], we aim to utilize deep learning (DL) models' classification capabilities to discern the nature of randomness in the synthetic bitstreams and thereby infer the masking of entropy source of the original RNG. DL's ability to identify dependencies in highly multivariate scenarios automatically leads to better non-interference detection accuracy than moment-based approaches using higher-order t-tests. This motivates our decision to incorporate a combination of GenAI-based entropy source modeling and DL-based non-interference test as a new RNG test to mitigate the shortcomings of NIST SP800-22 tests.

A. Our Contributions and Paper Outline

The key contributions of this paper are as follows:

1) **Low dimensional entropy source modeling:** Synthetic bitstreams generated via VAEs magnify the determinism in weak-entropy sources which are more likely to lie on lower-dimensional manifold as opposed to strong-entropy sources. This enables a clearer distinction between strong and weak entropy sources, which are otherwise hard to capture using NIST tests.
2) **Non-interference test to distinguish between entropy sources:** A DL based non-interference test is introduced

to assess the ability of synthetic bitstreams to mask sequences generated from two separable distributions. This test provides insights into the randomness properties of the synthetic bitstream, thereby enabling the evaluation of the entropy source properties of the RNG.

3) **Shorter Sequence Lengths:** GeNI achieves distinction between weak and strong entropy sources with significantly shorter sequences say, $\approx$ 100k bits, compared to the 1M sequence length required for NIST SP800-22 STS.
4) **Identification of weak entropy sources:** GeNI identifies sequences with artificial randomness, such as specious RNGs, which passes all NIST tests, by virtue of the inadequate masking capabilities of their synthetic bitstream. Moreover, other biased and correlated RNG sources are also recognized to possess weak entropy properties.

The rest of the paper is organized as follows. Sec. II entails the background knowledge of RNGs testing techniques and VAEs. Next, we delve into GeNI's entropy source modeling via latent space representation of VAEs in Sec. III. In Sec. IV, we present GeNI's non-interference testing. Sec. V describes the results of our proposed methodology on several case studies. Lastly, Sec. VI concludes the paper.

II. Background

A. Traditional RNG Evaluation Metrics

1) Entropy Estimates: To gauge the uniformity of the RNG sequences, we use two types of entropy estimates: a) a conservative estimate, i.e., min-entropy, and b) an accurate estimate, i.e., the Shannon entropy. They are calculated as:

$$\textbf{Min-entropy} = -\log_2 \max\{p_0, p_1\} \tag{1}$$

$$\textbf{Shannon-entropy} = -(p_0 \log_2\{p_0\} + p_1 \log_2\{p_1\}) \tag{2}$$

where p_0 and p_1 represent the probabilities of generating 0 & 1 by the RNG source. An equal probability of zeros and ones in the RNG leads to an entropy estimate of one in both cases.

2) Independence of the RNGs bits: In order to ensure that the current output has no influence on future outputs, we employ the auto-correlation test, which computes the correlation coefficient between the original sequence and lagging sequences [17]. Auto-correlation with lag k for a sequence of length N is defined as:

$$\text{ACF}(k) = \text{R}_{xx}(k) = \frac{\sum_{i=0}^{N-k-1}(x_i - \mu)(x_{i+k} - \mu)}{N\sigma^2} \tag{3}$$

where ACF(k) is the autocorrelation function for lag k, $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ is the RNG sequence and μ, σ^2 are the mean and the variance of the sequence.

3) Wide Sense Stationarity (WSS) property: WSS processes necessitate constancy in mean and auto-correlation across all samples, an important property for RNGs [17]. It is analyzed via the power spectral density (PSD) $\text{S}_{xx}(f)$ and its variance over all frequency windows.

$$\text{S}_{xx}(f) = \mathcal{F}(\text{R}_{xx}) = \sum_{i=-N+1}^{N-1} \text{R}_{xx}(k) e^{-2\pi f k i}, \tag{4}$$

where $\mathcal{F}(\text{R}_{xx})$ is the Fourier transform of the auto correlation function R_{xx}. We ascertain the WSS property by checking the variance of PSD to be less than a user-defined threshold σ_0.

B. Variational Auto-Encoder (VAE)

Variational Autoencoders (VAEs) stand as one of the foremost choices among generative AI model, distinct for their approach to modeling complex data distributions through a low dimensional latent space [11]–[13]. The probabilistic nature and the inclusion of the Kullback-Leibler (KL) divergence for regularization make VAEs particularly adept at producing varied and high-quality samples. Their application ranges widely, from enhancing image generation to facilitating the compression and learning of complex data representations. In this research, we leverage the capabilities of VAEs for a novel application: modeling the entropy sources underlying various RNGs, showcasing their versatility and efficacy in generative modeling.

III. GeNI's Low Dimensional Entropy Source Modeling

In the realm of RNG design and evaluation, entropy source modeling is crucial in directly influencing the integrity of cryptographic algorithms and protocols. TRNGs rely on entropy source modeling to capture the inherent physical properties that underpin randomness generation. Conversely, for PRNGs, entropy source modeling encompasses the characterization of seed entropy while considering any entropy loss introduced during the algorithmic process [18]. The absence of robust entropy source modeling stands out as a notable deficiency in the NIST SP 800-22 [10]. Understanding and accurately modeling entropy sources are paramount not only for evaluating the sufficiency of entropy provided by TRNGs but also for optimizing their throughput. For instance, the entropy and throughput of elementary oscillator-based TRNGs (EO-TRNGs) [19] hinge upon the low sensitivity of jitter sampling.

The intrinsic dimension of an RNG source, denoting the minimum number of independent states required for bitstream generation, holds significant sway in this context [20]. RNG sources with weak entropy exhibit a lower intrinsic dimension, which escalates as the entropy quality improves [21], [22]. In this context, the manifold hypothesis [15] is particularly useful for distinguishing between strong and weak entropy sources. RNGs with high intrinsic entropy produce uniformly distributed data points, which are challenging to represent on a low-dimensional manifold. Conversely, RNGs with lower intrinsic entropy exhibit more deterministic patterns, making their data points more likely to lie on a lower-dimensional manifold.

The low-dimensional modeling in our GeNI framework enables us to test different types of RNG sources as follows:

1) **"Bad" RNGs**: Weak entropy sources featuring low intrinsic dimensionality unveil hidden patterns in the bitstream, via low-dimensional modeling. This proves instrumental in magnifying deterministic or repeated patterns in RNGs with artificial randomness, such as specious RNGs [8], which may otherwise go unnoticed by NIST tests. Moreover, the low-dimensional latent distribution captures deterministic effects adeptly in RNGs with bias or correlation stemming from physical phenomena like process-voltage-temperature (PVT) variations or adversarial bias

injection, rendering hidden patterns more visible in the synthetic bitstream.

2) **"Good" RNGs**: The latent space representation by VAEs facilitates strong RNG entropy source modeling without necessitating specific knowledge of its underlying physical phenomenon. Stochastic models are relatively straightforward to construct for TRNGs characterized by high intrinsic dimensionality, though verifying all underlying physical assumptions poses challenges [23]. Similarly, PRNGs aim to rapidly accumulate entropy from diverse physical sources in the environment (e.g., keyboard presses, timing of interrupts) into the PRNG state, subsequently converting this high-entropy state into (pseudo) random bits [24] which are then used for synthetic bitstreams generation. In both cases VAEs generate bits from a low-dimensional representation of strong entropy source, contrasting with bit generation in weak entropy sources driven by hidden deterministic effects.

Thus, synthetic bitstreams generated via VAEs tend to magnify the determinism in weak-entropy sources more than in strong-entropy sources, highlighting the differences in entropy levels. However, to do so GeNI necessitates the following modifications to the conventional VAEs used for image generation: a) MLP encoder-decoder network architecture where the relationship of each bit to every other bit is better captured compared to a convolutional neural network (CNN) capturing the localized patterns in images, b) low-dimensional latent space representation to contrast between entropy sources, and c) non-interference testing to evaluate the quality of VAE output (described in Section IV). In this section, we first analytically demonstrate the representation prowess of a VAE using binary bit-streams which makes the latent parameters a strong function of the RNG entropy source. Thereafter, we elaborate our proposed synthetic data generation methodology using MLP architecture and the low-dimensional representation of the entropy source.

A. Latent Variables for a linear VAE

VAEs have been extensively prevalent in synthetic image and text generation [25] due to their ability to find latent, low-dimensional data representations. Each dimension of the latent space can represent a different aspect of the input data, such as pose, lighting, or identity in the case of images. We extend this representational ability of VAEs to RNG bitstreams, where the latent space maps to the properties of the entropy source, such as device noise statistics in the case of TRNG and seed entropy in the case of PRNGs. We analytically demonstrate that the latent space parameters for linear VAE are proportional to the statistical properties of the input bitstream. It is to be noted that practical VAEs use DNNs for encoders and decoders; we chose a linear model to demonstrate a mathematically tractable analysis.

Fig. 2 shows a VAE with a linear encoder and decoder, which takes in the RNG bitstream $\mathbf{x}$ as input and regenerates a synthetic bitstream $\mathbf{x}_s$. The RNG $\mathbf{x}$ is described by a parameterized probability distribution $p_\theta(x)$ where θ captures

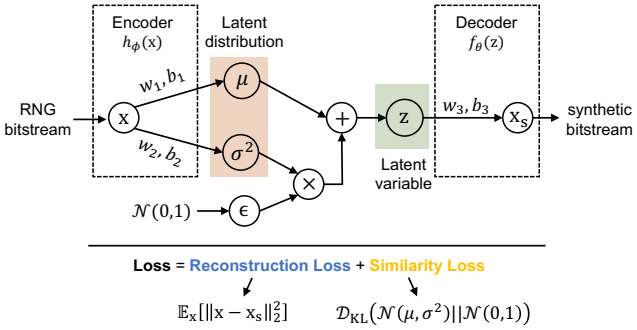

Fig. 2. VAE with linear encoders and decoders.

the entropy properties. For example, in the case of a Bernoulli random bitstream, $p_\theta(x)$ is given by:

$$p_\theta(x) = \theta^x \times (1-\theta)^{1-x}; x \in \{0,1\} \tag{5}$$

The objective of the VAE is to find the maximum likelihood estimate of the RNG distribution parameter θ. The maximum likelihood estimation involves maximizing the exact posterior probability $p_\theta(z|x)$, where z is the latent variable of the RNG. However, the maximum likelihood estimate is intractable to compute; therefore, VAEs approximate it using the posterior $q_\phi(z|x)$ closest to $p_\theta(z|x)$, where ϕ denotes the encoder parameters. The decoder parameterized by θ learns the conditional likelihood distribution $p_\theta(x|z)$ as f_θ regenerating an approximation of x, given by x_s. In this analysis, we consider unit-dimensional $x, z,$ and x_s for mathematical tractability. However, practical VAEs have $\dim(\mathbf{x}) = \dim(\mathbf{x}_s) > \dim(\mathbf{z})$, where $\mathbf{z}$ is the low dimensional latent representation.

The latent distribution is a Gaussian, i.e., $z \sim \mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2. The encoder denoted by $h_\phi(x)$ consists of two parts described as:

$$h_\phi(x) = h_\phi^{(1)}(x) + \epsilon \times h_\phi^{(2)}(x), \tag{6}$$

where $h_\phi^{(1)}(x) = \mu = w_1 x + b_1$ models the mean, $h_\phi^{(2)}(x) = \log(\sigma^2) = w_2 x + b_2$ models the variance (logarithm is chosen for numerical stability during training), ϵ is a sample from standard normal distribution $\mathcal{N}(0,1)$, and w_i's and b_i's are the weights and biases respectively. The well-known parametrization trick [11] is employed here to transform $\mathcal{N}(0,1)$ to latent variable z using the mean and standard deviation, such that $z = \mu + \epsilon \times e^{\log(\sigma^2)}$. Lastly, a linear decoder denoted by $f_\theta(z) = w_3 z + b_3$ is used to obtain the synthetic bitstream $\mathbf{x}_s$.

The loss function for VAEs reduces the KL divergence between the approximate posterior $q_\phi(z|x)$ and the exact posterior $p_\theta(z|x)$. Reducing the KL divergence is the same as maximizing the Evidence Lower Bound (ELBO) [11] given by:

$$\text{ELBO}(\theta, \phi; x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \text{KL}(q_\phi(z|x)||p(z)) \tag{7}$$

where $\mathbb{E}_{q_\phi(z|x)}[\cdot]$ denotes the expectation with respect to the approximate posterior $q_\phi(z|x)$, and $\text{KL}(q_\phi(z|x)||p(z))$ is the KL divergence between the approximate posterior and the prior $p(z)$ which is standard normal distribution.

TABLE I
ANALYTICAL EXPRESSION FOR THE VAE LOSS WITH LINEAR ENCODERS AND DECODERS.

$$\begin{aligned} L &= p_{\text{bit}}((1-w_3\cdot w_2)^2+(w_3\cdot w_1\cdot\sigma)^2-w_1^2+b_1\cdot(-w_2-w_3-2w_1)+w_3\cdot w_2^2\cdot b_1+w_3^2\cdot b_2\cdot w_2 \\ &\quad +2\cdot w_3^2\cdot w_1\cdot b_1+w_2-2b_3+2b_2\cdot w_3\cdot w_2)+w_3^2\cdot b_1\cdot b_2+(w_3\cdot b_1)^2 \\ &\quad +b_2-b_1^2-\exp(b_2)\cdot(1-p_{\text{bit}}+p_{\text{bit}}\cdot\exp(w_2))+b_3\cdot w_3\cdot b_2+b_3\cdot w_2\cdot b_1+b_3^2 \end{aligned} \tag{11}$$

$$\frac{\partial L}{\partial b_1} = \left(2w_3^2-2\right)b_1+\left(2p_{\text{bit}}w_1+b_2\right)w_3^2+\left(p_{\text{bit}}w_2^2-p_{\text{bit}}\right)w_3+\left(b_3-p_{\text{bit}}\right)w_2-2p_{\text{bit}}w_1=0 \tag{12}$$

$$\frac{\partial L}{\partial b_2} = \left(-p_{\text{bit}}\mathrm{e}^{w_2}+p_{\text{bit}}-1\right)\mathrm{e}^{b_2}+\left(p_{\text{bit}}w_2+b_1\right)w_3^2+\left(2p_{\text{bit}}w_2+b_3\right)w_3+1=0 \tag{13}$$

$$\frac{\partial L}{\partial b_3} = 2b_3+b_2w_3+b_1w_2-2p_{\text{bit}}=0 \tag{14}$$

$$\frac{\partial L}{\partial w_1} = 2p_{\text{bit}}\cdot\left(w_3^2-1\right)\left(w_1+b_1\right)=0 \tag{15}$$

$$\frac{\partial L}{\partial w_2} = -p_{\text{bit}}\mathrm{e}^{w_2+b_2}+p_{\text{bit}}\cdot\left(-2w_3\cdot(1-w_3w_2)+2b_1w_3w_2+b_2w_3^2+2b_2w_3-b_1+1\right)+b_1b_3=0 \tag{16}$$

$$\frac{\partial L}{\partial w_3} = p_{\text{bit}}\cdot\left(-2w_2\cdot(1-w_2w_3)+2b_2w_2w_3+2w_1^2w_3+4b_1w_1w_3+b_1w_2^2+2b_2w_2-b_1\right)+2b_1^2w_3+2b_1b_2w_3+b_2b_3=0 \tag{17}$$

The ELBO maximization is equivalent to the loss function minimization described by the encoder (ϕ) and decoder (θ) parameters as:

$$\begin{aligned} &\text{Loss}(\theta,\phi)=\frac{1}{n}\left(\sum_{i=1}^{n}\left(x_i-f_\theta\left(h_\phi\left(x_i\right)\right)\right)^2\right)+ \\ &\frac{1}{n}\times\frac{1}{2}\sum_{i=1}^{n}\left(1+h_\phi^{(2)}(x_i)-\left(h_\phi^{(1)}(x_i)\right)^2-e^{\left(h_\phi^{(2)}(x_i)\right)}\right), \end{aligned} \tag{8}$$

where the first term is the reconstruction loss, and the latter is the similarity loss. Expanding the reconstruction loss in terms of the VAE parameters $\phi=\{w_1,b_1,w_2,b_2\}$ and $\theta=\{w_3,b_3\}$ we get:

$$\begin{aligned} \text{Loss}_{\text{R}} &= \frac{1}{n}\left(\sum_{i=1}^{n}\left(x_i-f_\theta\left(h_\phi\left(x_i\right)\right)\right)^2\right) \\ &= \frac{1}{n}\left(\sum_{i=1}^{n}\left(x_i-w_3((w_1x_i+b_1)+(w_2x_i+b_2)\epsilon_i)-b_3\right)^2\right), \end{aligned} \tag{9}$$

where $\epsilon_i\sim\mathcal{N}(0,1)$ and $x_i\sim\text{Be}(p_{\text{bit}})$. The Bernoulli input distribution is the simplest approximation of an ideal RNG entropy source, where the bits are considered i.i.d. but with a probability of p_{bit}. Similarly, the similarity loss Loss_{sim} can be expressed as:

$$\begin{aligned} &\frac{1}{n}\times\frac{1}{2}\sum_{i=1}^{n}\left(1+h_\phi^{(2)}(x_i)-\left(h_\phi^{(1)}(x_i)\right)^2-e^{\left(h_\phi^{(2)}(x_i)\right)}\right) \\ &=\frac{-1}{2n}\sum_{i=1}^{n}\left(1+(w_2x_i+b_2)^2-(w_1x_i+b_1)^2-e^{w_2x_1+b_2}\right). \end{aligned} \tag{10}$$

The only random variables in the loss function are x_i and e_i, whose sample means can be replaced using their expected values for large n, where n denotes the number of bits in the VAE training bitstream. The replacement of sample means via the expected values considerably simplifies the loss expression, making the analysis more tractable, providing insights into the dependence of the optimal VAE parameters

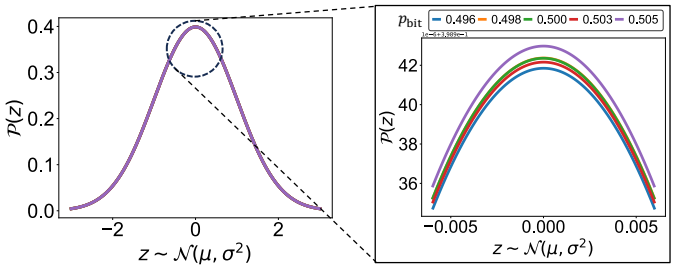

Fig. 3. Latent distributions across different entropy parameters p_{bit}.

(μ,σ^2) on the entropy source parameters (p_{bit}). Furthermore, the expected values of the terms with ϵ in them going to zero, i.e., $\mathbb{E}[\epsilon],\mathbb{E}[x\epsilon],\mathbb{E}[x^2\epsilon]=0$ and $\mathbb{E}[\epsilon^2]=1$ as they are sampled from the standard Gaussian distribution $\mathcal{N}(0,1)$. On the other hand, the expected values of x terms are well known for the Bernoulli distribution with both the $\mathbb{E}[x]=\mathbb{E}[x^2]=p_{\text{bit}}$. Substituting the expected values of the RVs in Eq. (9) and (10), we obtain a closed-form expression for the total loss given by Eq. (11). The algebraic expression for loss can be differentiated with respect to the VAE parameters to obtain Eq. (12), (13), (14), (15), (16), (17), with six variables i.e. $\{b_1,b_2,b_3,w_1,w_2,w_3\}$. This set of equations, as shown in Table. I, can be solved using optimization methods such as gradient descent.

Figure 3 shows the drift in the latent distribution $\mathcal{N}(\mu,\sigma^2)$ w.r.t. the change in entropy parameters the RNG, i.e., p_{bit}. The noticeable separation between the distributions signifies the ability of VAEs to abstract the properties of the entropy source in its latent distribution. Minimizing the KL divergence with respect to the standard normal, both the (μ,σ^2) parameters are close to $(0,1)$. However, their exact values depend on the entropy parameter p_{bit}, as minimizing the loss function in Eq. 10 also requires us to minimize the reconstruction loss. This leads to a noticeable vertical distribution drift as we move away from the ideal value of 0.5 (zoomed-in plot of Fig. 3).

We extend the idea of latent space representation of entropy source to practical VAEs with deep neural networks as encoders and decoders, generating a synthetic bitstream from the RNG data. One of the critical limitations associated with VAEs

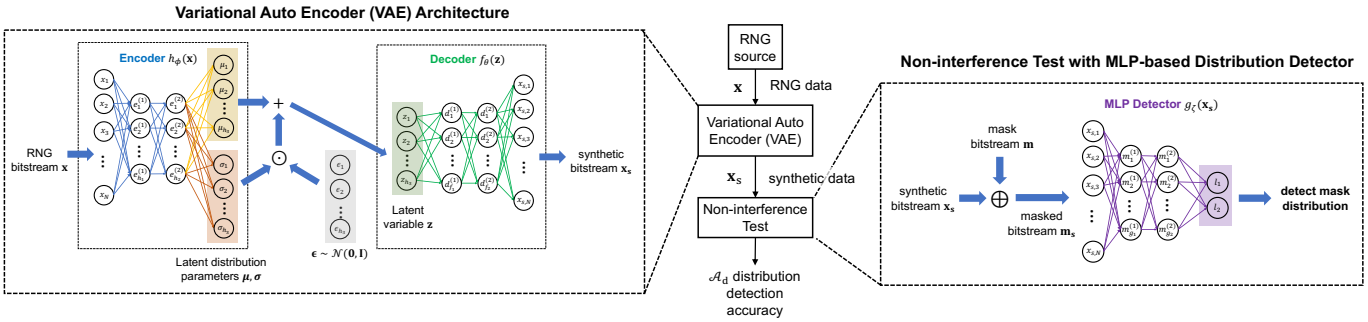

Fig. 4. ML-based randomness testing including VAE for generating synthetic bitstream $\mathbf{x}_s$, and non-interference test.

is the proximity of the latent and input distributions, which affects probabilistic sample generation. This proximity is influenced by both the latent space dimension and the encoder-decoder architecture. Interestingly, for different RNGs with varying levels of entropy and intrinsic dimensions, the VAE output can distinguish between different entropy sources by employing a low latent space dimension. Thus, we transform a potential drawback into a strategic advantage for randomness testing by using a low-dimensional representation of entropy sources. The synthetic bitstream amplifies the deterministic properties of the RNG entropy source, making it easier to distinguish the quality of randomness. Typically, the low-dimensional latent space is multivariate and cannot be visually distinguished for different entropy sources. Therefore, in the next section, we employ the principle of non-interference testing to distinguish between the synthetic bitstreams for different RNGs generated from the latent distribution.

Algorithm 1: Synthetic bitstream generation using VAE

Training phase inputs: RNG bitstream $\mathbf{x}_{\text{train}}$
/* VAE with MLP-based encoder and decoder structure defined in Listing 1 */
model = VAE(input-size, latent-size)
epoch = 0
while *epoch* $<$ *num-epoch* **do**
  recon$_x$, mean, logvar = model($\mathbf{x}_{\text{train}}$)
  model.train(recon$_x$, mean, logvar, $\mathbf{x}_{\text{train}}$)
  /* train using ADAM to minimize reconstruction and similarity loss */
  epoch = epoch + 1
Generation phase inputs: RNG bitstream $\mathbf{x}_{\text{test}}$
/* synthetic bitstream generation */
$\mathbf{x}_s$ = model($\mathbf{x}_{\text{test}}$)

B. Synthetic Data Generation

In order to model the RNG entropy source, we learn the parameters of a latent space representation of the RNG using a VAE with multi-layer perception (MLP) based encoders and decoders as shown in Fig. 4. The RNG bitstream is broken into N-bit chunks, fed as the input to the first layer of the encoder, followed by two hidden layers of sizes h_1, h_2 respectively. Thereafter, the last layer of the encoder maps to the multivariate latent space parameters of $\boldsymbol{\mu}$ and the $\log\left(\boldsymbol{\sigma}^2\right)$ each with h_3 dimensions. These latent space parameters are reparametrized with an independently sampled standard multivariate Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$ to obtain the latent variable samples $\mathbf{z}$ as follows:

$$\mathbf{z} = \boldsymbol{\mu} + e^{\log(\boldsymbol{\sigma}^2)} \odot \boldsymbol{\epsilon}, \tag{18}$$

where $\odot$ is the element-wise multiplication operation between two vectors $e^{\log(\boldsymbol{\sigma}^2)}$ and $\boldsymbol{\epsilon}$. The decoder generates a N-dimensional synthetic bitstream that is interleaved over multiple runs to obtain $\mathbf{x}_s$.

```
class VAE(nn.Module):
    def __init__(self, input_size, latent_size):
        super(VAE, self).__init__()

        # Encoder
        self.fc_mean = nn.Linear(mlp_h1(input_size), latent_size)
        self.fc_logvar = nn.Linear(mlp_h2(input_size), latent_size)
        # Decoder
        self.fc = nn.Linear(mlp_f(latent_size), input_size)

    def encode(self, x):
        mean = self.fc_mean(nn.reLU(mlp_h1(x)))
        logvar = self.fc_logvar(nn.reLU(mlp_h2(x)))
        return mean, logvar

    def reparameterize(self, mean, logvar):
        std = torch.exp(0.5 * logvar)
        eps = torch.randn_like(std)
        return mean + eps * std

    def decode(self, z):
        recon_x = nn.Sigmoid(nn.reLU(mlp_f(z)))
        return recon_x

    def forward(self, x):
        mean, logvar = self.encode(x)
        z = self.reparameterize(mean, logvar)
        recon_x = self.decode(z)
        return recon_x, mean, logvar
```

Listing 1. VAE for synthetic bitstream generation

C. VAE Network Architecture

We utilize MLP networks for both the encoder and decoder stages of the VAE, as illustrated in Listing 1. During training, the VAE aims to minimize two key losses over num-epochs iterations using the ADAM optimizer: a) the reconstruction loss, quantifying the disparity between the synthetic output

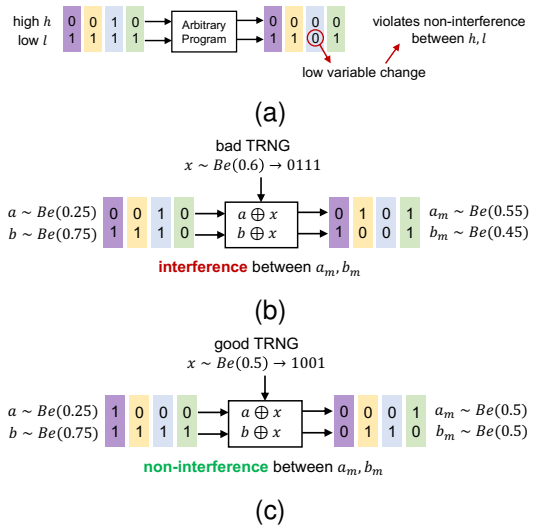

Fig. 5. (a) Principle of non-interference in general; Applying non-interference theory for sequences (b) with distinguishable and (c) indistinguishable randomness

generated by the VAE and the training bitstream $\mathbf{x}_{\text{train}}$ utilized to produce it, and b) the similarity loss, measuring the difference between the distributions $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ and the prior standard distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$, as depicted in Algorithm 1. Both the encoder (responsible for both mean and variance) and decoder networks consist of identical MLPs comprising two hidden layers each. The output dimensions of the encoder and the input dimension of the decoder are set to be identical, corresponding to the latent dimension of the VAE. Subsequently, the trained model is employed to generate the synthetic bitstream $\mathbf{x}_s$ from a test RNG bitstream $\mathbf{x}_{\text{test}}$. In the next section, we use the non-interference testing to distinguish between the strength of the entropy sources via the synthetic bitstreams generated from their latent space representation.

IV. GENI'S NON-INTERFERENCE TESTING

In the realm of synthetic bitstream generation using Variational Autoencoders (VAEs), we evaluate randomness through the lens of non-interference testing. At its core, interference signifies the disclosure of sensitive information, where a high variable serves as the source and a low variable as the target. This principle, introduced in [16], dictates that sensitive information should remain concealed. Non-interference implies that if program executions with identical low inputs but different high inputs produce varying low outputs, information about the high inputs has leaked (Fig. 5a).

Our randomness evaluation centers on the analysis of non-interference within masked sequences. These sequences are generated by XORing synthetic bitstreams with deterministic sequences $\mathbf{a}$ and $\mathbf{b}$ of having low correlation. A detection test discerns between two distributions $\mathbf{a}_m = \mathbf{a} \oplus \mathbf{x}_s$ and $\mathbf{b}_m = \mathbf{b} \oplus \mathbf{x}_s$, where $\mathbf{x}_s$ is the synthetic bitstream generated by the VAE. A deep learning binary classifier evaluates the randomness level in $\mathbf{x}_s$ using the distribution detection accuracy $\mathcal{A}_d$, discerning between low (Fig.5b) and high (Fig.5c) randomness scenarios.

We first establish the criteria that we develop to evaluate the quality of the randomness of the tested sources. First, let us define the notion of randomness formally: A generator $G(x)$ is said to qualify as a random number generator if there does not exist any efficient algorithm, say A, which can distinguish the output of $G(x)$ and U, where U is a uniform distribution, with a probability more than $\frac{1}{2}+\epsilon$, where ϵ is a non-negligible quantity.

Now, we state the test formally as follows:

Lemma IV.1. *A sequence $\mathbf{x}_s$ is drawn from a random number generator (RNG) if and only if there does not exist any efficient algorithm to distinguish between the masked bitstreams $\mathbf{a}_m = \mathbf{a} \oplus \mathbf{x}_s$, and $\mathbf{b}_m = \mathbf{b} \oplus \mathbf{x}_s$, where $\mathbf{a}$ and $\mathbf{b}$ are drawn from two distinct distributions.*

Proof. The sufficient condition is trivial. If $\mathbf{x}_s$ is sampled from U, then it is straight-forward to see both $\mathbf{a}_m$ and $\mathbf{b}_m$ are drawn from U too. Wlog for a one bit value $a_m \in \mathbf{a}_m$, if $x_s \in \mathbf{x}_s$, st. $Pr[x_s = 1] = Pr[x_s = 0] = \frac{1}{2}$, $\Rightarrow Pr[a_m = a \oplus x_s = 1] = Pr[a = 1].Pr[x_s = 0] + Pr[a = 0].Pr[x_s = 1] = \frac{1}{2}(Pr[a = 0] + Pr[a = 1]) = \frac{1}{2}$. Likewise, $Pr[a_m = 0] = \frac{1}{2}$ too, and hence they are indistinguishable.

The necessary condition, can be proved by the contrapositive statement, that if $\mathbf{x}_s$ is not sampled from $U \Rightarrow$ there exists an efficient algorithm B to distinguish between $\mathbf{a}_m$ and $\mathbf{b}_m$, given that $\mathbf{a}$ and $\mathbf{b}$ are drawn from two distinct distributions. The definition of B can be stated by using A, which can distinguish $\mathbf{x}_s$ and U. B takes the input $\mathbf{a}_m$, and computes $\mathbf{a}_m \oplus \mathbf{a} = \mathbf{x_s}$. On the other hand, it takes the input $\mathbf{b}_m$, and computes $\mathbf{b}_m \oplus u$, where $u \in U$. By the proof of the sufficient condition, $\mathbf{b}_m \oplus u \in U$. Hence, now A can be used to distinguish $\mathbf{x}_s$ from U, in turn leading B to efficiently distinguish $\mathbf{a}_m$ and $\mathbf{b}_m$. □

```
model = nn.Sequential(
  nn.Linear(input_size, 1024),
  nn.BatchNorm1d(1024),
  nn.ReLU(),
  nn.Linear(1024, 128),
  nn.BatchNorm1d(128),
  nn.ReLU(),
  nn.Linear(128, 2),
  nn.Softmax())
```

Listing 2. MLP architecture for non-interference testing

A. Deep Net Architecture for Detecting Non-interference

Deep Net's ability to represent abstract data makes it immensely valuable for classification tasks. For non-interference testing using DL models, a higher accuracy indicates that the masked bitstreams are distinguishable or the synthetic bitstream has low randomness. The average inference accuracy of a K-fold cross-validation framework indicates the masking prowess of the synthetic bitstreams and, thereby, the quality of the residual entropy sources.

Our approach to randomness testing with DL involves an iterative process to optimize dataset sizes for training. Given the absence of a definitive guideline for determining the appropriate amount of training data in DL, we start with reasonably small training and inference sets, taking cues from previous applications of non-interference testing [26]. These sizes are then incrementally adjusted until a user-defined dataset size limit is reached, which reveals the minimum number of samples required for detecting non-interference.

Algorithm 2: Non-interference Testing using MLP

Input: synthetic bitstream $\mathbf{x}_s$
Output: distribution detection accuracy $\mathcal{A}_d$
$\mathbf{a}_m = \mathbf{a} \oplus \mathbf{x}_s$, $\mathbf{a} \sim \text{Be}(0.25)$
$\mathcal{T}_a = \langle \mathbf{a}_m, \mathbf{0} \rangle$
$\mathbf{b}_m = \mathbf{b} \oplus \mathbf{x}_s$, $\mathbf{b} \sim \text{Be}(0.75)$
$\mathcal{T}_b = \langle \mathbf{b}_m, \mathbf{1} \rangle$
$\mathcal{D} := \mathcal{T}_a \cup \mathcal{T}_b$;
$\boldsymbol{\mathcal{A}}_d = \{0\}, s = s_{\text{init}}, t = 0$
while $s < S$ **do**
  `/* MLP classifier from Listing 2 */`
  model = MLPClassifier(input-size)
  $\mathcal{D}_t = \texttt{SubSample}(\mathcal{D}, s)$ `/* select s/2 samples from` $\mathcal{T}_a$ `and` $\mathcal{T}_b$ `*/`
  $\langle \mathcal{D}_t^1, \mathcal{D}_t^2, \ldots, \mathcal{D}_t^K \rangle := \texttt{GenCrossValidSet}(\mathcal{D}_t)$
  epoch = 0; i = 0;
  while $i < K$ **do**
    model.ResetWeights()
    $\mathcal{T}_{i,\text{train}} := \cup_{j=0, j\neq i}^{K} \mathcal{D}_t^j$; $\mathcal{T}_{i,\text{test}} := \mathcal{D}_t^i$
    while *epoch* $<$ *num-epoch* **do**
      `/* train using ADAM to minimize cross-entropy loss */`
      model.train($\mathcal{T}_{i,\text{train}}$)
      epoch = epoch + 1
    `/* Classification stage */`
    $\boldsymbol{\mathcal{A}}_d[t]$ = $\boldsymbol{\mathcal{A}}_d[t]$ + model.evalAcc($\mathcal{T}_{i,\text{test}}$) ;
    i = i + 1
  $\boldsymbol{\mathcal{A}}_d[t] = \boldsymbol{\mathcal{A}}_d[t]/K$ `/* Average` $\boldsymbol{\mathcal{A}}_d[t]$ `*/`
  $s = s + s_{\text{init}}; t = t + 1$

Note that once the size of the data set is known, the model is trained once, and inference is performed multiple times using the trained model for a given RNG.

The DL-based leakage assessment experiment is outlined in Algorithm 2, where the complete dataset $\mathcal{D}$ under consideration is a union of two subsets: $\mathcal{T}_a \cup \mathcal{T}_b$. The masked instances from $\mathcal{T}_a$ are labeled as 0, while those from $\mathcal{T}_b$ are labeled as 1. The training and inference process commences with a modest dataset $\mathcal{D}_t$ of size s_{init} obtained by subsampling equal constituents of $\mathcal{T}_a, \mathcal{T}_b$ from the complete dataset $\mathcal{D}$, denoted by the `SubSample` in Algorithm 2. The size of the dataset $\mathcal{D}_t$ is progressively expanded in each iteration by adding an equal number of samples from both constituent sets. For cases involving high entropy RNGs, an optimization strategy is proposed to streamline the learning process. If the classification accuracy saturates below $< 100\%$ with the maximum sample size (e.g., S = 10^7), a final confirmation test is conducted with a larger sample size (e.g., 5×10^7 samples), ensuring reliability in test results.

To ensure the robustness of our training and validation procedures, especially when dealing with small datasets, we employ the stratified K-fold cross-validation method [29]. This approach is widely recognized for its effectiveness in preventing overfitting. The K-fold cross-validation operates by randomly dividing the entire dataset $\mathcal{D}_t$ into K equal-sized subsets, denoted as $\mathcal{D}_t^1, \mathcal{D}_t^2, \ldots, \mathcal{D}_t^K$. The stratified nature of this partitioning ensures that each subset $\mathcal{D}_t^j$ contains an equal number of samples from both classes (label-0 and label-1) as implemented by the `GenCrossValidSet` function in Algorithm 2. Subsequently, K - 1 of these subsets is utilized for training the model $\mathcal{M}$, while the remaining subset serves as the inference or test set. This process is iterated K times, allowing each subset to be used as the inference set once. The primary objective is to assess whether the model $\mathcal{M}$ can effectively generalize its learned knowledge to unseen datasets. The distribution detection accuracy is computed for different dataset sizes (indexed by t) after taking the average $\boldsymbol{\mathcal{A}}_d[t]$ over K-subsets. Next, we test GeNI for different types of RNGs with varying strengths of entropy sources.

V. Comparison of different RNGs using GeNI

In this section, we present results for the conventional randomness testing techniques, followed by our proposed randomness testing methodology on different types of RNGs. We use VAE parameters of input-size of 8192, $h_1 = 1024, h_2 = 512$, and latent-size of 16. On the other hand, MLP for non-interference follows the parameters of Listing 2, with an input size of 32, unless otherwise specified. Thereafter, we analyze the dependence of the model parameters of the distribution detection accuracy $\mathcal{A}_d$.

A. Case Studies for Different RNGs

In this subsection, we elaborate on the limitations of traditional randomness evaluation methods, such as min-entropy, Shannon entropy, autocorrelation function (ACF) for assessing correlation, and the analysis of stationarity through the variance of the power spectral density (PSD) function, in accurately assessing the randomness characteristics of RNGs.

1) TERO TRNG: Table II shows that RNG data obtained from a TERO TRNG implementation [23], [30] on Artix-7 FPGA passes all NIST tests. In Fig. 6a, the speckle pattern corresponding to a subset of the RNG bitstream $\mathbf{x}$ is depicted, revealing a min-entropy of "0.99990327" and a Shannon-entropy of "0.99999999" over a sample set of 100M samples. However, the synthetic bitstream $\mathbf{x}_s$, generated from the TERO TRNG data using a VAE, exhibits subpar statistical properties due to the low-dimensional representation of the entropy source. While both $\mathbf{x}$ and $\mathbf{x}_s$ display consistent autocorrelation properties over varying lags, the correlation for $\mathbf{x}_s$ surpasses that of $\mathbf{x}$ owing to reduced entropy. Similarly, the stationarity analysis reveals that while the variance of the power spectral density for both $\mathbf{x}$ and $\mathbf{x}_s$ falls within an expected range, the latter exhibit higher variance. Moving forward, we leverage the conventional randomness testing results for TERO TRNG as a benchmark for evaluating the efficacy of these testing schemes for Specious Randomness and PRNG sequences.

2) Mersenne Twister Based PRNG: The Mersenne Twister (MT) based PRNGs are widely utilized for generating random numbers in popular Python libraries such as NumPy [27] and Linux libraries due to their efficient implementation and strong randomness properties. Our conventional randomness tests confirm this, with the MT-PRNG successfully passing all NIST tests and exhibiting comparable performance to TERO TRNG in terms of min-entropy, Shannon-entropy, autocorrelation, and

TABLE II
NIST SP800-22 AND GENI TEST RESULTS FOR DIFFERENT TYPES OF RNGS

	TERO TRNG [23]		Mersenne Twister PRNG [27]		CSPRNG [28]		Low bias CSPRNG [28]		Low Correlation CSPRNG [28]		Specious RNG [8]		Memory Requirement (GB)	Execution Time (seconds)
Entropy Source	Strong						Weak				Weak†			
NIST Tests	**P-values**	**Result**	**P-values**	**Result**	**P-values**	**Result**	**P-values**	**Result**	**P-values**	**Result**	**P-values**	**Result**		
Frequency	0.657933	PASS	0.4486	PASS	0.1981	PASS	0	FAIL	0.7061	PASS	1*	PASS	1.1	6.52
Block Frequency	0.935716	PASS	0.3798	PASS	0.7514	PASS	0	FAIL	0.5461	PASS	0.4321	PASS	1.1	7.28
Cumulative Sums	0.419021	PASS	0.5927	PASS	0.3214	PASS	0	FAIL	0.7820	PASS	0.8341	PASS	0.98	176
Runs	0.096578	PASS	0.4664	PASS	0.0931	PASS	0	FAIL	0	FAIL	0.9016	PASS	1.1	16.74
Longest Run	0.4934392	PASS	0.2566	PASS	0.2123	PASS	0.8088	PASS	0	FAIL	0.8066	PASS	1.1	0.054
Rank	0.085587	PASS	0.1329	PASS	0.3078	PASS	0.7918	PASS	0.4905	PASS	0.3206	PASS	1.1	112.41
FFT	0.383827	PASS	0.7089	PASS	0.9943	PASS	0.0210	PASS	0	FAIL	0.5320	PASS	1.1	72.34
Non Overlapping Template	0.616305	PASS	0.7411	PASS	0.9895	PASS	1	PASS	1	PASS	0.3401	PASS	1.1	21.64
Overlapping Template	0.574903	PASS	0.3518	PASS	0.5591	PASS	0.0563	PASS	0	FAIL	0.2127	PASS	1.09	0.197
Universal	0.236810	PASS	0.3737	PASS	0.0158	PASS	0	FAIL	0	FAIL	0.7379	PASS	1.09	13.27
Approximate Entropy	0.574903	PASS	0.7496	PASS	0.1579	PASS	0	FAIL	0	FAIL	0.9999	PASS	0.98	1777
Random Excursions	0.980883	PASS	0.0571	PASS	0.0848	PASS	0.1327	PASS	0	FAIL	0.0946	PASS	0.98	176
Random Excursions Variant	0.585209	PASS	0.01933	PASS	0.1878	PASS	0.0845	PASS	0.333	PASS	0.0118	PASS	0.99	30
Serial	0.574903	PASS	0.4806	PASS	0.1541	PASS	0	FAIL	0	FAIL	0.9999	PASS	0.98	1725
Linear Complexity	0.122325	PASS	0.0282	PASS	0.3052	PASS	0.298	PASS	0.3748	PASS	0.2736	PASS	1.1	5560
NIST SP800-22 Evaluation	PASS						FAIL				PASS†		1.1&	5560
GeNI Accuracy	59.8%		61.28%		60.22%		79.8%		69.89%		72.58%		3.3#	335
GeNI Evaluation	PASS						FAIL				FAIL†			

∗ specious RNG [8] has exactly equal number of ones and zeros; † Specious RNG has artificial randomness but passes all the NIST SP800-22 tests, whereas our proposed GeNI captures its inherent compressibility [8]; & Memory access for NIST SP800-22 is for the test with the worst case execution time, i.e., Linear Complexity Text including the RNG data; # Memory access on case GeNI involves VAE, MLP parameters for non-interference, and RNG data.

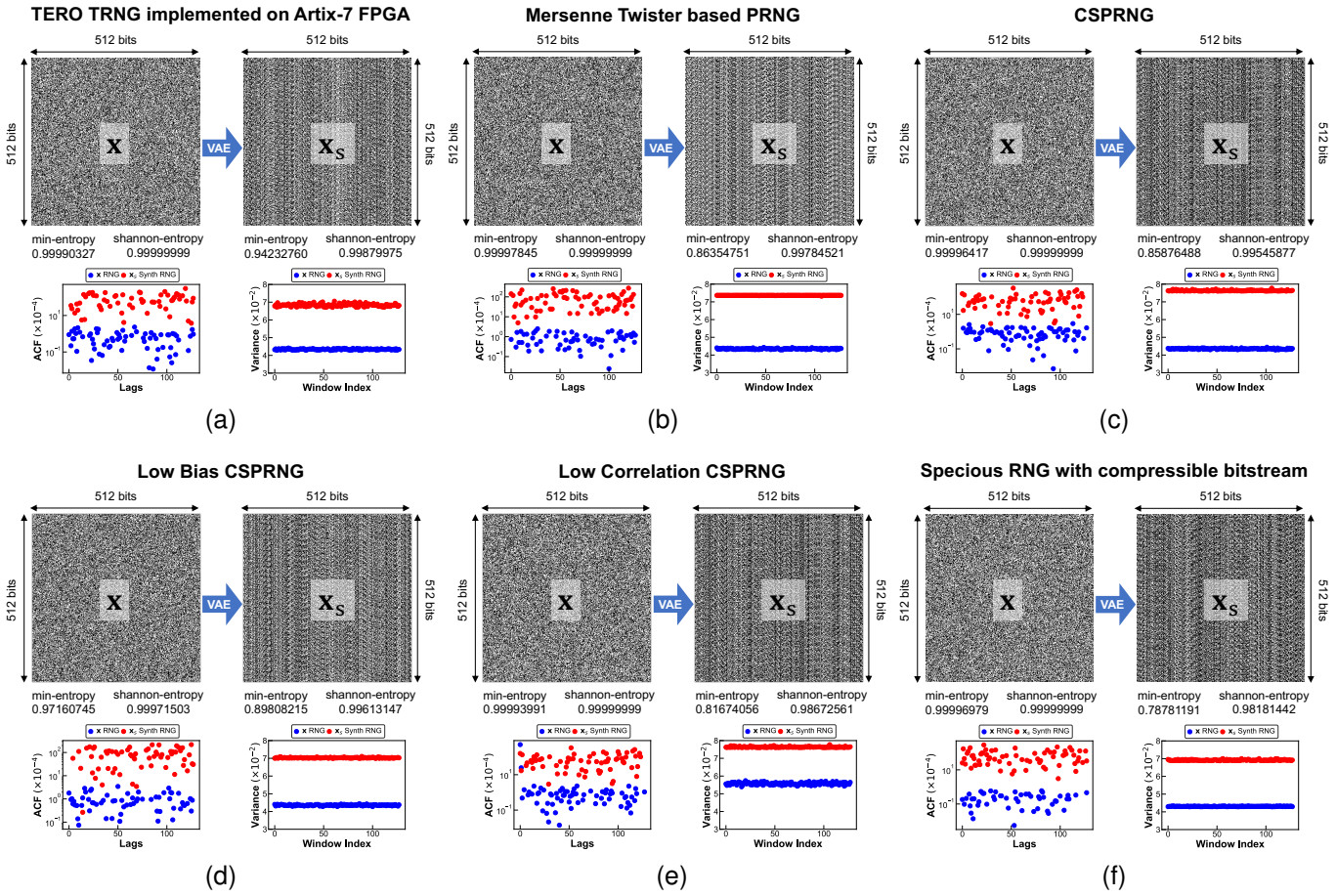

Fig. 6. The speckle pattern (Black = "1", White = "0"), autocorrelation function (ACF), and variance of the power spectral density (PSD) as a measure of stationarity for the RNG bitstream $\mathbf{x}$ and the synthetic bitstream $\mathbf{x}_s$ generated from VAE for (a) TERO TRNG [23], (b) Mersenne Twister based PRNG [27], (c) Biased CSPRNG [28], (d) Correlated CSPRNG [28], (e) CSPRNG [28], and (f) Specious RNG [8]. It may be noted that these results signify that the conventional metrics are necessary but not sufficient for evaluating randomness.

stationarity properties, as depicted in Table II and Fig. 6b. However, the synthetic sequence $\mathbf{x}_s$ demonstrates a slightly lower min-entropy of "0.86354751" and Shannon-entropy of "0.99784521" than that of the TERO TRNG. Consequently, further randomness testing using non-interference tests is deemed necessary.

3) CSPRNG: To thoroughly assess our proposed randomness testing suite for PRNGs with weak entropy, we conducted experiments on a cryptographically secure (CS) PRNG [28], including its biased and correlated variants. While the pure CSPRNG successfully passes all NIST tests, as evidenced

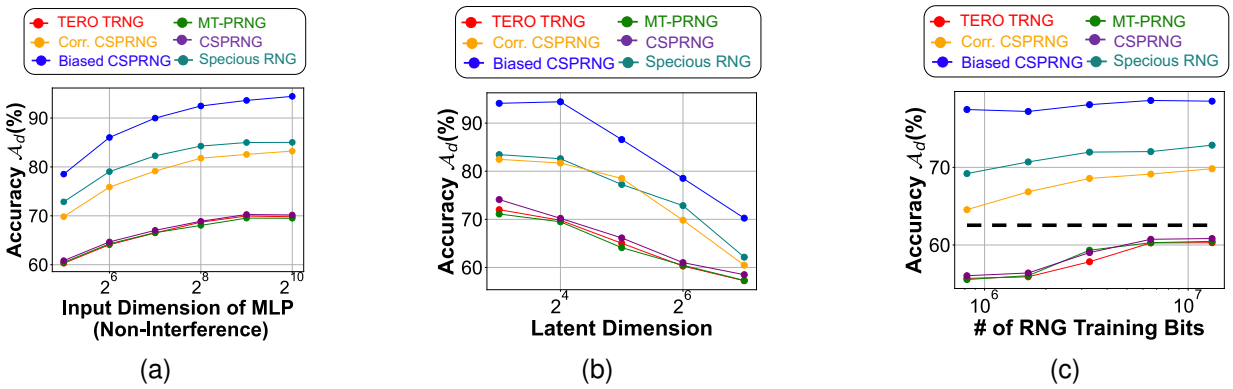

Fig. 7. Distribution detection accuracy $\mathcal{A}_d$ of the masked distribution generated from the synthetic bitstream $\mathbf{x}_s$ vs. (a) input dimension of MLP in the non-interference stage with VAE latent dimension of 16, (b) latent dimension of VAE for MLP input dimension of 128, and (c) the number of training bits for the RNG with MLP input dimension of 32.

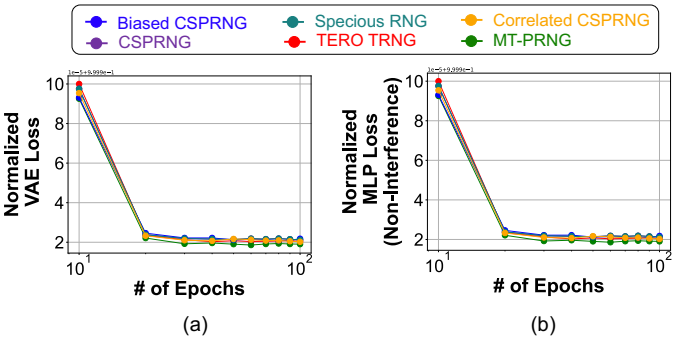

Fig. 8. Loss function vs. number of epochs due training for (a) VAE, including the reconstruction loss and similarity loss, and (b) cross entropy loss in MLP for the non-interference stage in the case of different types of RNGs.

in Table II, it encounters failures upon introducing a bias of 0.51 (zero bias condition of 0.5) and correlation of 0.02 (zero correlation condition of 0). Notably, the biased-induced CSPRNG (Fig. 6d) exhibits a decrease in min-entropy and Shannon-entropy compared to the pure CSPRNG (Fig. 6c) and the correlated CSPRNG (Fig. 6e). Interestingly, while the autocorrelation and stationarity properties remain largely unchanged upon the addition of bias (Fig. 6d), they exhibit an increase in magnitude with the incorporation of correlation in the bitstream (Fig. 6e). Similar to the previous observations for TRNG and specious RNG, the synthetic bitstreams $\mathbf{x}_s$ have poor statistical properties compared to original bitstream $\mathbf{x}$.

4) Specious Randomness: The artificial patterns created by shuffling all elements of the 24-bit binary number system exhibit specious randomness [8], contrasting with genuine randomness obtained from an entropy source. Surprisingly, these specious RNGs pass all NIST tests, as indicated in Table II, and demonstrate similar min-entropy, Shannon-entropy, autocorrelation, and stationarity properties compared to the TRNG, as illustrated in Fig. 6f. However, the low-dimensional modeling of the entropy source diminishes the min-entropy to "0.78781191" and Shannon-entropy to "0.98181442", capturing only the randomness inherent in the ordering of the 24-bit binary numbers in the concatenated bitstream. Although the autocorrelation function (ACF) and power spectral density (PSD) variances resemble the synthetic TRNG sequence, further randomness testing using non-interference is warranted. These findings underscore the necessity of adopting a more comprehensive approach to randomness testing, which includes the modeling of the entropy source, as a single metric may not fully capture all relevant information.

B. Accuracy vs. Input Dimension of MLP (Non-Interference)

Figure 7(a) illustrates the effectiveness of GeNI in detecting masked distributions particularly, by applying it to the synthetic sequences derived from all the six randomness sources discussed previously in Table II with NIST SP 800-22 tests. As discussed in Sec. IV we present the detection accuracy $\mathcal{A}_d$ for the masked distributions. We observe high distribution detection accuracy ($\mathcal{A}_d$) for correlated PRNGs, biased RNGs, and specious RNGs at lower input dimensions for MLPs when the size of the training RNG set is fixed to 10^6 (includes both VAE and MLP for non-interference). Note that training is done once with a fixed RNG dataset to obtain the model parameters. The accuracy saturates as the input dimension of MLP or the size of the first layer increases. This increase indicates that a successful distribution detection is possible at a low MLP input dimension of 32, amounting to ≈100k RNG bits, $10\times$ lower than the $1M$ bits typically needed for NIST tests. RNGs with high internal entropy exhibit lower distribution detection accuracy as they resist low dimensional entropy source modeling. Conversely, weaker entropy sources enable low-dimensional modeling achieved through VAEs capturing entropy source properties with higher fidelity and resulting in higher $\mathcal{A}_d$. Note that specious RNGs characterized by compressible bitstreams demonstrate higher $\mathcal{A}_d$ even when they successfully pass all NIST tests. This underlines the superior quality of the proposed tests in identifying weaknesses in a given candidate RNG.

C. Accuracy vs. Latent Space Dimension of VAE

We determine the optimal choice for the latent dimension of the Variational Autoencoder (VAE) through empirical experiments. As illustrated in Fig. 7(b), diminishing the latent dimension enhances the discrimination in non-interference testing accuracy until reaching a saturation point below 16 dimensions. For weak entropy sources, lower latent dimensions increase accuracy, indicative of a pronounced deterministic influence within the synthetic bitstream. Conversely, enlarging the latent dimension diminishes both accuracy and the distinguishability between strong and weak entropy sources,

as augmenting the model's complexity better captures the inherent characteristics of RNGs in synthetic outputs. In our case, a latent dimension of 16 adequately represents effectively evaluating and contrasting the performance across TRNGs, PRNGs, and specious RNGs.

D. Accuracy vs. Training Set Size of Non-Interference Testing

Figure 7(c) demonstrates that the accuracy in detecting distributions reaches a plateau with increased training dataset size. Initially, the training process involves both the VAE for synthesizing data and the MLP for conducting non-interference testing, utilizing inputs from various RNGs as outlined in Algorithms 1 and 2. The detection accuracy, denoted as $\mathcal{A}_d$, stabilizes once the dataset size exceeds 10^6 bits, indicating a common saturation point applicable across different RNG categories. Furthermore, the loss functions of both the VAE (Fig. 8(a)) and MLP (Fig. 8(b)) models decline over successive training epochs with a slight but noticeable divergence in the final VAE loss values. This hints at the relationship between the entropy source and the VAE model parameters, consistent with our latent distribution analysis for VAEs with different $p_{\mathbf{bit}}$ in Fig. 3.

E. GeNI Test PASS/ FAIL Criterion

To determine an accuracy threshold for the PASS/FAIL criteria using GeNI, we selected the most parameter-efficient testing conditions: an input dimension of 32 for the MLP in the encoder/ decoder architecture, a latent dimension of 16, and a minimum of 10^6 training samples for non-interference testing. Under, these assumptions, a dotted line in Fig. 7(c) marks the minimum accuracy (65%) needed to identify non-random patterns within a synthetic bitstream. This threshold is used to determine the PASS/ FAIL criteria of RNG sources, as shown in Table II. However, this accuracy limit will vary with different parameter choices for the MLP architecture and latent dimension, as illustrated in Fig. 7 (a) and (b). The limit increases with the input dimension and decreases with the latent dimension. Adjusting the parameters of GeNI can be useful for determining the point of failure of an RNG source whose entropy is affected by PVT variations or adversarial noise. All our experiments were conducted on an Intel(R) Xeon(R) Gold 6226 CPU @ 2.70 GHz with 96 cores, 2 threads per core, 12 cores per socket, and 256GB DRAM. Table II shows that the execution time for GeNI is 16.5 times faster compared to NIST, whose execution time is determined by the linear complexity test requiring the maximum time. Additionally, the memory access for GeNI includes VAE, non-interference MLP model parameters, and the RNG data, leading to a slightly higher overhead compared to the NIST test, which only requires the RNG data.

F. Accuracy vs. Entropy

To gain deeper insights into the dependence of GeNI accuracy on the strength of the entropy source, we varied the correlation coefficient and bias for a CSPRNG, effectively altering the strength of its entropy source. Figure 9 demonstrates that as correlation and bias in the bitstream increase, the accuracy improves while the min-entropy decreases. The correlation is a less effective tuning parameter for reducing overall entropy

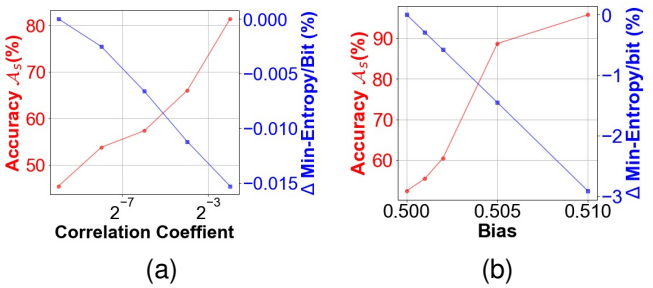

Fig. 9. GeNI accuracy and min-entropy trends vs. (a) correlation coefficient, and (b) bias for CSPRNG.

TABLE III
r-ROBUSTNESS OF VAEs FOR TERO TRNG FOR $r = 30$, FOR DIFFERENT LATENT SPACE DIMENSIONS, AND NOISE LEVELS.

Noise Type		No Noise	Bias			
$p_{\text{n-bias}}$			0.25	0.5	0.75	1
Latent Dimension	8	0.534	0.21	0.01	0	0
	16	0.560	0.28	0.05	0	0
	32	0.553	0.29	0.07	0	0
	64	0.556	0.31	0.07	0	0
	128	0.545	0.32	0.08	0	0

compared to bias, as indicated by the percentage change in min-entropy in Fig. 9. This result highlights the strong relationship between the strength of the entropy source and the classification accuracy in our proposed randomness testing scheme. It is important to note that altering the entropy for a TRNG would require circuit modifications, whereas modifying the entropy for MT-PRNGs and specious RNGs would involve algorithmic changes. The dependence of GeNI accuracy on the entropy of these RNGs necessitates a focus on both design and analysis, presenting a promising direction for future research.

G. Reliability and Robustness of VAEs

To assess robustness, we evaluated the impact of noise $\boldsymbol{\delta}$ added to the RNG $\mathbf{x}$. This was done by observing the change in the synthetic bitstream $\mathbf{x}_s$, upon noise injection. We modeled the noise in the case of RNGs as bias, where each RNG bit is set to '1' with a Bernoulli probability $p_{\text{n-bias}}$. When $p_{\text{n-bias}} = 1$, the bitstream consists entirely of '1's and, $p_{\text{n-bias}} = 0$ represents a no-noise scenario. Bias injection can occur due to process-voltage-temperature variations or intentional noise injection by an adversary.

We then evaluated the r-robustness of the VAE [31], measured by the probability that the change in the synthetic bitstream $\mathbf{x}_s$ is less than r (in terms of the L2 norm), given by:

$$\Pr(\|f_\theta(h_\phi(\mathbf{x}+\boldsymbol{\delta},\boldsymbol{\epsilon}_1)) - f_\theta(h_\phi(\mathbf{x},\boldsymbol{\epsilon}_2))\|_2 < r), \qquad (19)$$

where $f_\theta(h_\phi(.))$ is the VAE, $\boldsymbol{\delta}$ is the noise, $\boldsymbol{\epsilon}_1, \boldsymbol{\epsilon}_2$ are the standard Gaussian noise samples following (6) and r is the robustness bound. The value of r is chosen to be 30, matching the mean value of the L2 norm of the difference in the synthetic bitstream $\mathbf{x}_s$ in the no-noise scenario, i.e., $\mathbb{E}[\|f_\theta(h_\phi(\mathbf{x},\boldsymbol{\epsilon}_1)) - f_\theta(h_\phi(\mathbf{x},\boldsymbol{\epsilon}_2))\|_2]$.

Table III lists the r-robustness values of a VAE with an input sequence length of 8192 for TERO TRNG with $r = 30$. Probability values greater than 0.5 indicate strong robustness for the VAE in all scenarios [31]. In the case of bias injection with $p_{\text{n-bias}} > 0.5$, we observe r-robustness drop to 0, since

the L2 norm exceeds the threshold of $r = 30$, approaching the limiting value of $\sqrt{8192 \times 0.5} = 64$ when the noisy $\mathbf{x}_s$ approaches an all '1's state, compared to $\mathbf{x}_s$ in the no-noise case with 50% 1s and 0s. This robustness analysis confirms the reliability of the test, i.e., the ability of the test to distinguish between the original and the perturbed scenario is very high.

VI. Conclusion

This is the first work that explores the use of GeNI, generative AI-based non-interference testing techniques, to address the key shortcomings of the popularly employed NIST SP 800-22 testing suite: the lack of entropy source modeling. We employ a variational autoencoder (VAE) with a low-dimensional latent space representation to contrast the entropy of different RNGs. The synthetic bitstreams generated via VAEs accentuate the deterministic attributes of weak entropy sources, which are more likely to reside on a low-dimensional manifold compared to strong entropy sources. The subsequent non-interference detection step evaluates the masking capability of these synthetic bitstreams, achieving higher accuracy in distinguishing weak-entropy sources from strong ones. Additionally, our proposed GeNI operates with bitstreams of $\approx$ 100k in length instead of the 1M bits used for NIST tests and achieves $16.5\times$ faster execution time. GeNI can detect the quality of the entropy source, flagging artificial sources of randomness, such as specious RNGs, which pass all NIST tests.

References

[1] Y. Yu *et al.*, "Can deep learning break a true random number generator?" *IEEE TCAS II: Express Briefs*, vol. 68, no. 5, pp. 1710–1714, 2021.

[2] John Markoff, "Flaw found in an online encryption method," https://web.archive.org/web/20170215005520/http://www.nytimes.com/2012/02/15/technology/researchers-find-flaw-in-an-online-encryption-method.html, 2012.

[3] Mike Bendel, "Hackers Describe PS3 Security As Epic Fail, Gain Unrestricted Access," https://www.exophase.com/20540/hackers-describe-ps3-security-as-epic-fail-gain-unrestricted-access/, 2010.

[4] National Institute of Standards and Technology (NIST), "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," https://csrc.nist.gov/pubs/sp/800/22/r1/upd1/final, 2010.

[5] ——, "Public comments on sp 800-22 rev. 1a, a statistical test suite for random and pseudorandom number generators for cryptographic applications," https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/initial-comments/sp800-22r1a-initial-public-comments-2021.pdf, 2021.

[6] M.-J. O. Saarinen, "SP 800–22 and GM/T 0005–2012 Tests: Clearly Obsolete, Possibly Harmful," in *IEEE EuroS&PW*, 2022, pp. 31–37.

[7] Y. Dodis *et al.*, "Security analysis of pseudo-random number generators with input: /dev/random is not robust," in *ACM SIGSAC CCS*, 2013.

[8] J. Almlöf *et al.*, "Creating and detecting specious randomness," *EPJ Quantum Technology*, vol. 10, no. 1, p. 1, 2023.

[9] B. Yang *et al.*, "TOTAL: TRNG on-the-fly testing for attack detection using lightweight hardware," in *IEEE DATE*, 2016, pp. 127–132.

[10] National Institute of Standards and Technology (NIST), "Decision to Revise NIST SP 800-22 Rev. 1a," https://csrc.nist.gov/news/2022/decision-to-revise-nist-sp-800-22-rev-1a, 2022.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[12] D. P. Kingma *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, 2019.

[13] W. Liu *et al.*, "Towards visually explaining variational autoencoders," in *Proceedings of the IEEE CVPR*, 2020, pp. 8642–8651.

[14] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE CVPR*, 2023, pp. 4195–4205.

[15] "Representation learning: A review and new perspectives," *IEEE transactions on PAMI*, 2013.

[16] D. Clark, S. Hunt, and P. Malacaria, "Quantified interference: Information theory and information flow," in *WITS*, 2004.

[17] J. J. M. Chan *et al.*, "Ensuring quality of random numbers from trng: Design and evaluation of post-processing using genetic algorithm," *Journal of Computer and Communications*, 2016.

[18] F. Dörre and V. Klebanov, "Practical detection of entropy loss in pseudo-random number generators," in *ACM CCS*, 2016, pp. 678–689.

[19] V. Fischer and D. Lubicz, "Embedded evaluation of randomness in oscillator based elementary trng," in *CHES 2014*. Springer.

[20] W. H. Zurek, "Algorithmic randomness and physical entropy," *Physical Review A*, vol. 40, no. 8, p. 4731, 1989.

[21] J. B. Tenenbaum *et al.*, "A global geometric framework for nonlinear dimensionality reduction," *science*, 2000.

[22] D. Zhao *et al.*, "Latent variables on spheres for autoencoders in high dimensions," *arXiv preprint arXiv:1912.10233*, 2019.

[23] P. Haddad *et al.*, "A physical approach for stochastic modeling of tero-based trng," in *CHES 2015*. Springer, pp. 357–372.

[24] S. Coretti *et al.*, "Seedless fruit is the sweetest: Random number generation, revisited," in *Annual International Cryptology Conference*. Springer, 2019, pp. 205–234.

[25] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.

[26] S. Saha *et al.*, "Leakage assessment in fault attacks: a deep learning perspective," *Cryptology ePrint Archive*, 2020.

[27] NumPy Developers, "Random generator," https://numpy.org/doc/stable/reference/random/generator.html#numpy.random.Generator.

[28] David Johnston, "A python implementation of the sp800-22 rev 1a prng test suite." https://github.com/dj-on-github/sp800_22_tests, 2021.

[29] X. Zeng *et al.*, "Distribution-balanced stratified cross-validation for accuracy estimation," *Journal of Experimental & Theoretical Artificial Intelligence*, 2000.

[30] K. Pratihar *et al.*, "Birds of the same feather flock together: A dual-mode circuit candidate for strong PUF-TRNG functionalities," *IEEE TC*, 2022.

[31] A. Camuto *et al.*, "Towards a theoretical understanding of the robustness of variational autoencoders," in *ICAIS*. PMLR, 2021, pp. 3565–3573.

Kuheli Pratihar is a PhD student at the Department of Computer Science and Engineering, IIT Kharagpur under the supervision of Prof. Debdeep Mukhopadhyay and Prof. Rajat Subhra Chakraborty. Her primary research work involves design, lightweight implementation, and analysis of hardware security primitives like Physically Unclonable Functions (PUFs), True Random Number Generators (TRNGs) etc..

Rajat Subhra Chakraborty is a professor with the Department of Computer Science and Engineering, IIT Kharagpur. His area of research is hardware security, VLSI design and digital content protection through watermarking. He is a senior member of the IEEE and ACM.

Debdeep Mukhopadhyay received his PhD degree from IIT Kharagpur. Currently, he is a full professor with the Department of Computer Science and Engineering, IIT Kharagpur. His research interests include hardware security, micro-architectural attacks, cryptography, VLSI. He is a senior member of IEEE and ACM, and the recipient of the prestigious Shanti Swarup Bhatnagar prize 2021.