Enhancing Prediction Entropy Estimation of RNG for On-the-Fly Test

Yuan Ma[1,2], Weisong Gu[1,2], Tianyu Chen[1*], Na Lv[1], Dongchi Han[1,2], and Shijie Jia[1,2]

[1] SKLOIS, Institute of Information Engineering, CAS, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{mayuan, guweisong, chentianyu, lvna, handongchi, jiashijie}@iie.ac.cn

Abstract. Random number generators (RNGs) play a vital role in cryptographic applications, and ensuring the quality of the generated random numbers is crucial. At the same time, on-the-fly test plays an important role in cryptography because it is used to assess the quality of the sequences generated by entropy sources and to raise an alert when failures are detected. Moreover, environmental noise, changes in physical equipment, and other factors can introduce variations into the sequence, leading to time-varying sequences. This phenomenon is quite common in real-world scenarios, and it needs on-the-fly test. However, in terms of speed and accuracy, current methods based on mathematical formulas or deep learning algorithms for evaluating min-entropy both fail to meet the requirements of on-the-fly test. Therefore, this paper introduces a new estimator specifically designed for on-the-fly min-entropy estimation. To accurately evaluate time-varying data, we employ an appropriate change detection technology. Additionally, we introduce a new calculation method to replace the original global prediction probability calculation approach for accuracy. We evaluate the performance of our estimator using various kinds of simulated datasets, and compare our estimator with other estimators. The proposed estimator effectively meets the requirements of on-the-fly test.

Keywords: On-the-fly test · Entropy estimation · Prediction estimator · Change detection technology · Confidence interval.

1 Introduction

In today's cryptographic engineering applications, random numbers have become increasingly important. For instance, in key distribution and mutual authentication schemes, two communicating parties collaborate to exchange information for key distribution and authentication purposes. These random numbers are generated by random number generators that contains entropy sources, and entropy sources are divided into two categories: stationary sources and time-varying sources. Secure random numbers are often used as security primitives for many

* Corresponding author: chentianyu@iie.ac.cn.

cryptographic applications, so it is necessary to evaluate the quality of random numbers.

Current methods for evaluation are mainly divided into two categories: white-box test and black-box test. White-box test, also known as theoretical entropy evaluation, requires an understanding of the internal structure and generation principle of the entropy source. It establishes a mathematical model according to appropriate assumptions to calculate the theoretical entropy of the output sequence [8]. However, given the complex and varied structures of many entropy sources, it becomes challenging to model them accurately, thereby limiting the applicability of theoretical entropy evaluation. Black-box test includes statistical test and statistical entropy evaluation: statistical test uses hypothesis-testing methods to conduct tests on the sequence for some properties, determining whether the tested sequence meets the null hypothesis (indicating randomness) or exhibits statistical defects [13]. Nevertheless, it is worth noting that certain specifically constructed pseudo-random sequences may exhibit favorable statistical properties and successfully pass these tests, posing potential security threats. Statistical entropy does not require the knowledge of the internal structure and generation principle of entropy sources. It evaluates the safety of the random numbers from the perspective of "entropy" [15]. In summary, to meet the requirements of generality and security, statistical entropy evaluation has become an indispensable approach.

Statistical entropy evaluation methods can be categorized into two main categories: those based on mathematical and statistical theories, and those based on deep learning. However, some estimators in the former, represented by the NIST SP800-90B standard, have been found to have overestimation and underestimation problems when faced with some typical datasets during entropy evaluation [21]. The latter has a problem of high time consumption. They both don't perform well in time-varying sequence which is common in reality. Thus, in order to detect RNG failures quickly and reliably, we need an on-the-fly test that is suitable for time-varying datasets.

To design a suitable estimator for on-the-fly test, we need solve two issues. Firstly, as mentioned above, we should update the model in a timely manner, especially for time-varying datasets. To address it, we utilize the change detection technique. Secondly, we introduce a new calculation method for global predictability of entropy estimation [15], specifically designed to handle situations involving small samples or extreme probabilities (i.e., probabilities approaching 0 or 1), which is different from the SP800-90B Standard, because the raw method is no longer suitable for on-the-fly test.

Our goal is to design an entropy estimator which meets the requirement of speed and accuracy for on-the-fly test. We present several significant contributions in this paper:

1) We propose a modified version of the prediction estimators from SP800-90B, enabling an on-the-fly test for evaluating the quality of entropy sources timely. To support the new framework, we proposed two key technologies: change detection technique and new calculation method for global predictability.

2) By leveraging the characteristics of the prediction estimator model and drawing inspiration from neural network parameter adjustments during training, we design a novel change detection technique suitable for online entropy estimation. Besides, we are the first to address the challenges associated with evaluating min-entropy in scenarios involving small sample datasets and extreme probabilities. We provide a reasonable solution to this issue, which plays a critical role in on-the-fly test.

3) We compare the performance among our estimator and other existing estimators, using different types of simulated datasets with known entropy values. The experimental results show that, our estimator performs well for all different types of tested datasets, outperforming the other ones.

The rest of this paper is organized as follows. In Section 2, we introduce the definition of min-entropy, along with an overview of the 90B standard. In Section 3, we expound and analyze the existing estimators. Section 4 presents our new framework and provides detailed descriptions, including the change detection technique and so on. In Section 5, we present a series of experiments comparing our estimator with other estimators. Finally, in Section 6, we conclude our paper.

2 Preliminaries

2.1 Min-Entropy

"Entropy" is the unit representing the size of information in communication, which can quantify the randomness of the output sequence [15]. Min-entropy is a conservative way to ensure the quality of random numbers in the worst case. The definition of min-entropy is as follows: we take the next output from an entropy source as a random variable X, which is an independent discrete random variable. If X takes value from the set $A = \{x_1, x_2, ..., x_k\}$ with probability $Pr\{X = x_i\} = p_i$ for $i = 1, ..., k$, the min-entropy of the output is

$$\mathrm{H_{min}} = \mathrm{min}_{1\leq \mathrm{i} \leq \mathrm{k}}[\text{-}\mathrm{log_2(p_i)}] = \text{-}\mathrm{log_2}[\mathrm{max}_{1\leq \mathrm{i} \leq \mathrm{k}}(\mathrm{p_i})] \,. \tag{1}$$

If the min-entropy of X is H, then the probability of any value that X can take doesn't exceed 2^{-H}. For a random variable with the possibility of k distinct values, the maximum value that the min-entropy can reach is $log_2 k$, achieved when the variable follows a uniform probability distribution, i.e., $p_1 = p_2 = ... = p_k = 1/k$.

2.2 NIST SP800-90B Standard

The 90B estimation suite is a widely-used standard for calculating statistical entropy [15]. It calculates global predictability and local predictability with an upper bound of 99% confidence, and chooses the maximum value between them to estimate min-entropy. The suite comprises ten distinct entropy estimators that will be discussed in Section 3.

Global Predictability: Global predictability is the proportion of all predicted data to be correctly predicted. For a given prediction method, let $p'_{global} = c/n$, where c represents the number of correct predictions and n denotes the number of predictions made. Then, to give a conservative calculation method, 90B calculates p_{global} according to the following equation [7]:

$$p_{global} = \begin{cases} 1 - 0.01^{1/n}, & p'_{global} = 0 \\ \min(1, p'_{global} + 2.576\sqrt{\frac{p'_{global}(1-p'_{global})}{n-1}}), & otherwise \end{cases}, \tag{2}$$

which is the upper bound of the 99% confidence interval on p'_{global}, and it should meet the condition of De Moivre-Laplace Central Limit Theorem, that is: let $X_1, X_2, \ldots, X_n$ be i.i.d Bernoulli random variables with success probability $p \in (0,1)$ such that $np \to \infty$, as $n \to \infty$. Denote $S_n : X_1 + X_2 + ... + X_n$ and

$$Y_n^* = \frac{S_n - np}{\sqrt{np(1-p)}}.$$

Then, $\forall\ y \in R$, the theorem states that

$$lim_{n\to\infty}[P(Y_n^* \le y)] = \Phi(y) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y} e^{-t^2/2}\, dt. \tag{3}$$

Local Predictability: Local predictability is based on the longest run of correct predictions, which is valuable mainly when the source falls into a state of very predictable output for a short time [4]. Let l be the number one larger than the longest run of correct predictions. Then local predictability is calculated as

$$0.99 = \frac{1 - p_{local}x}{(l+1-lx)q} \cdot \frac{1}{x^{n+1}}, \tag{4}$$

where $q = 1 - p_{local}$, n represents the number of predictions, and x is the real positive root of the equation $1 - x + qp_{local}^{l}x^{l+1} = 0$. Then by iterations and the binary search, we can solve the mentioned equation and calculate the local predictability.

3 Related Work

3.1 Statistical Entropy Evaluation

Statistical entropy evaluation is comprised of estimators based on statistic methods and deep learning algorithms. For the former, in 2018, the final NIST SP800-90B test suite published, which is a typical representative of the statistical entropy estimations, which is based on min-entropy and specifies how to design and test entropy sources.It employs ten different estimators to calculate the min-entropy [15]. While it performs well on stationary datasets, it falls short when dealing with time-varying datasets. Before conducting entropy estimation, the 90B standard carries out an initial IID (independent and identically distributed) test. If the dataset meets the IID requirement, the MostCommon Estimator is utilized. Otherwise, the suite employs ten different estimators and selects the minimum value among them. These ten estimators can be divided into two categories: statistic-based and prediction-based. On the one hand, statistic-based estimators treat the test sequence as a whole and employ statistical methods to analyze properties related to entropy sources. On the other hand, prediction-based estimators use a training set comprised of previously observed samples to predict the next sample. By comparing the predicted results with the actual samples, the success rate of prediction is determined, and entropy estimation is performed based on the probability of successful prediction. Prediction-based estimators have a better performance than the other estimators in this standard. A brief introduction of the 10 estimators is as follows.

−**Most Common Value Estimator** performs entropy estimation based on the frequency of the most commonly occurring sample values in the sequence.

−**Collision Estimator** performs entropy estimation based on the collision frequency of samples in the sequence.

−**Markov Estimator** assumes the sequence as a first-order Markov process for entropy estimation.

−**Compression Estimator** is an entropy estimator based on the Maurer's algorithm.

−**T-Tuple Estimator** calculates entropy based on the occurrences of some fixed length repeated tuples.

−**LRS Estimator** calculates entropy based on the occurrences of some longer repeated tuples.

−**MultiMCW Prediction Estimator** utilizes four sliding windows of different sizes to determine the most frequently occurring value for prediction. A scoreboard is employed to determine the appropriate sliding window to use.

−**Lag Prediction Estimator** selects a prediction period ranging from 1 to 128 and also employs a scoreboard to select the optimal period.

−**MultiMMC Prediction Estimator** begins by setting up a dictionary (a two-dimensional array) and a scoreboard (a one-dimensional array). The dictionary is responsible for counting the frequency of prefixes and suffixes, while the scoreboard keeps a record of accurate predictions. After counting, it calculates the min-entropy.

–**LZ78Y Prediction Estimator** creates a dictionary based on patterns observed in the sequences and uses it for prediction.

Table 1. 90B Estimators.

Statistic-based	**Prediction-based**
MostCommon Value Estimator	MultiMCW Prediction Estimator
Collision Estimator	Lag Prediction Estimator
Markov Estimator	MultiMMC Prediction Estimator
Compression Estimator	LZ78Y Prediction Estimator
T-Tuple Estimator	
LRS Estimator	

For the latter, Yang et al. [20] were the first to apply neural networks to entropy source evaluation in 2018. In 2020, Lv et al. [12] conducted a comprehensive study on parameter settings for fully-connected neural networks (FNN) and recurrent neural networks (RNN), achieving accurate estimates of M-sequences with up to 20 stages. In 2019, Zhu et al. [21] combined change detection techniques with neural networks, partially resolving the issue of inaccurate prediction for time-varying sequences, and their model is named CDNN. Furthermore, in 2023, Zhang et al. [10] utilized TPA-LSTM to quantify the unpredictability of random numbers, and validated the effectiveness of pruning and quantized deep learning models in the field of random number security analysis. The above methods provide increasingly accurate estimation, but the speed needs to be improved.

In summary, the prediction-based estimators of SP800-90B can provide the same accurate estimation as the deep learning based estimators for stationary datasets and some time-varing datasets, and the former can consume less time.

3.2 On-the-fly Test Technologies

In terms of the on-the-fly test applied in cryptography, Santoro et al. [14] conducted the evaluation of the harmonic series on FPGA in the entropy test in 2009. Then, in 2012, Veljković et al. [16] proposed the online implementation for NIST SP800-22 and Yang et al. [18] improved it in 2015.

At the same time, Yang et al. [19] completed hardware implementations of 4 statistic-based estimators of NIST SP800-90B on FPGA after some simplifications, but it is only aimed at the estimators of the first draft 90B and its accuracy needs to be improved. In 2017, Grujić et al. [6] used the three prediction estimator of NIST SP800-90B to implement the on-the-fly test, but the results is not very accurate because there some mistakes in the second draft standard, and besides, the latest draft is also not suitable because the dictionaries updates laggardly. Then, in 2021, Kim et al. [9] proposed an online estimator that updates the min-entropy estimate as a new sample is received, which is based on the idea of the compression estimate of NIST SP800-90B, and it is implement on software. However, it doesn't perform well in time-varying datasets, even in some stationary datasets. Therefore, new framework should be designed to improve it.

4 New Framework of the 90B's Prediction Estimator for On-the-fly Test

4.1 Design Goal and Principle

Our design goal and principle is to achieve on-the-fly test effectively, so we need to improve speed while ensuring accuracy. We have tested that the minimum of time consumption of estimators based on deep learning is 30 seconds for processing 1Mbit of data which can't meet the requirement of on-the-fly test. By contrast, the raw 90B estimators only consume 0.15 seconds. Therefore, we design the new framework according to the 90B estimators.

Besides, we know that on-the-fly test requires as few estimators as possible to reduce the time consuming, and prediction estimators outperform the other ones [7]. Therefore, while ensuring accuracy, we choose the four prediction estimators included in 90B to modify for on-the-fly test. Last but not least, suitable change detection technology and calculation method of global predictability should be designed to improve the accuracy.

4.2 Framework of Our Estimator

We can observe that the predictors in SP800-90B all feature scoreboards or dictionaries, which serve as key components in the prediction process. However, the estimation accuracy of these predictors in handling time-varying sequences is compromised. This can be attributed to the fact that, even as the datasets change, the scoreboards and dictionaries retain information from the previous datasets. As a result, there is a lag in the response of the dictionaries and scoreboards to data changes during accumulation, leading to prediction errors when applied to new datasets. Therefore, it is imperative to make improvements in this regard.

We have made the following modifications to the aforementioned estimators for conducting on-the-fly test. The entire process is presented in Figure 1. In it, *point* is the change position, and i is the serial number of the sample. For each estimator, we perform the simultaneous operations of reading in data and outputting results in a serial manner. In step one, considering that the dictionaries and scoreboards have not yet started accumulating data at startup, which may result in erroneous estimation, we exclude the first 4999 samples from undergoing entropy estimation. During this phase, only the dictionaries and scoreboards are accumulated. Then, in the second step, at the point when there are 5000 samples, we calculate the prediction probability as the initial value for the change detection process based on the accumulated dictionaries. Starting from the 5001st

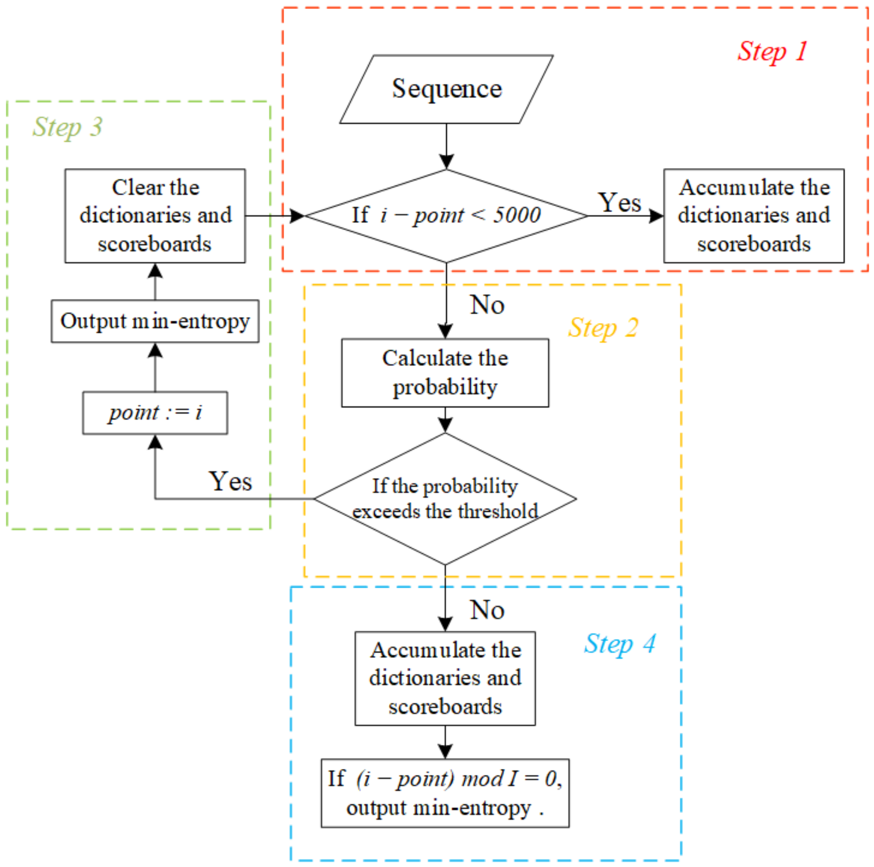

Step 1
Sequence
If $i - point < 5000$
Yes
Accumulate the dictionaries and scoreboards
Step 3
Clear the dictionaries and scoreboards
Output min-entropy
$point := i$
Yes
No
Step 2
Calculate the probability
If the probability exceeds the threshold
No
Step 4
Accumulate the dictionaries and scoreboards
If $(i - point) \bmod I = 0$, output min-entropy .

Fig. 1. The flowchart of the new framework.

sample, we calculate the prediction probability for each subsequently read-in sample. This calculated probability serves as the basis for the change detection technology.

In step three, if the prediction suddenly deviates and exceeds the threshold, we output the calculated min-entropy, clear the dictionaries and scoreboards, and initiate a new round of entropy estimation. Otherwise, as shown in the fourth step, if there is no change appearing, for every I samples input, entropy calculation is performed according to the formula in Section 4.4, and the results are outputted without clearing the dictionaries and scoreboards. Throughout this process, the minimum value among the four estimators is selected as the final output result. Here, I refers to the interval between two outputs.

4.3 Change Detection Module

In Figure 1, we employ a sequential approach for the four estimators to carry out the accumulation of dictionaries and scoreboards. We then utilize change detection technology to identify changes in the datasets and promptly clear the dictionaries and scoreboards when such changes are detected. This is followed by initiating a new round of dictionaries accumulation and scoreboards counting.

The current change detection technology can be categorized into three types: error rate-based drift detection, data distribution-based drift detection, and multiple hypothesis test drift detection [11]. The latter two methods require more time and resource consumption as they involve additional feature extraction and comparison processing on the data. Consequently, they are not suitable for our on-the-fly test scenario. Error rate-based drift detection, specifically the widely used Drift Detection Method (DDM) [5], offers a viable approach. Its concept is as follows: when the sample dataset exhibits stable distribution, the error rate of the model gradually decreases with the input of data; when there is a change in the probability distribution, the error rate of the model increases. We can reference the DDM approach for our change detection modules, but some adjustments will be necessary in terms of specific details.

For our estimator, when the probability distribution changes, the error rate of our model has a possibility of both an increase and a decrease; when the sample dataset is stable, the error rate is in an almost stable and unchanging state. Besides, Gama et al. [5] set the confidence level for drift to 99%, and the drift level is reached if $p_i + s_i \geq p_{min} + 3 \times s_{min}$, where p_i is the probability corresponding to the first i samples and s_i is the standard deviation of the first i samples. p_{min} is the minimum probability of the previous samples and s_{min} is the corresponding standard deviation. However, the inequality can hold only when the normal approximation of the binomial distribution holds. Therefore, we replace it with another formula for general which mentioned in Section 4.4, that is,

$$p_i \geq \frac{p_{min} + \frac{z^2_{\alpha/2}}{2(i_{min}-1)} + z_{\alpha/2}\sqrt{\frac{p_{min}(1-p_{min})}{i_{min}-1} + \frac{z^2_{\alpha/2}}{4(i_{min}-1)^2}}}{1 + \frac{z^2_{\alpha/2}}{i_{min}-1}}, \tag{5}$$

and for the lower bound of confidence interval, we use the formula

$$p_i \leq \frac{p_{max} + \frac{z^2_{\alpha/2}}{2(i_{max}-1)} - z_{\alpha/2}\sqrt{\frac{p_{max}(1-p_{max})}{i_{max}-1} + \frac{z^2_{\alpha/2}}{4(i_{max}-1)^2}}}{1 + \frac{z^2_{\alpha/2}}{i_{max}-1}}, \tag{6}$$

where p_{max} is the maximum probability of the previous samples and $z_{\alpha/2}$ is 2.576 when the confidence level is 99%. This can apply to all situations.

Then, during the estimation, we use the prediction probabilities of four estimators for simultaneous change detection. As long as a probability that exceeds the confidence interval appears, it is determined that a change has occurred and immediately clear the dictionaries and scoreboards. In theory, our method is similar to the hyperparameter update of deep learning algorithms, but more timely than it.

4.4 Optimization of Global Predictability for Small Sample Datasets and Extreme Probability

According to Section 2.2, the calculation method of global predictability confidence interval is divided into two situations. When p'_{global} is zero, it uses Clopper-Pearson Exact Method [3]. In other cases, use normal distribution to approximate binomial distribution and calculate the confidence interval.

However, the above method only contains situation when $np > 5$ and $n(1 - p) > 5$, or $p = 0$ or $p = 1$, where n is the sample size and p is the probability. Therefore, we should consider the case that $np \leq 5$ or $n(1-p) \leq 5$ to make the perfect.

In our proposed online estimator, the dataset was truncated according to the sample distribution and parameter changes due to the use of change detection technology. Besides, in the process of entropy estimation, the probability may approach to 0 or 1. The two factors may make us encounter the case mentioned above, i.e., $np \leq 5$ or $n(1-p) \leq 5$. In this case, the confidence interval calculation of the global prediction probability in 90B standard is no longer valid because it does not meet the condition that the binomial distribution is approximated to normal distribution, that is, the central limit theorem mentioned in Section 2.2. Therefore, we need to use a new method to calculate it.

Poisson approximations can do for the above issue to some extent, but it doesn't provide the method of calculating the confidence interval [4]. T-distribution can also handle some small sample issues, but still can't solve the above problem completely [2]. Therefore, we use "Plus Four Confidence Intervals" to handle the small sample issue here. This is proposed by Edwin Bidwell Wilson in 1927, which is an asymmetric interval [17]. It can be used for any probability value between 0 and 1 in the case of the small sample securely. It is obtained by solve the equation of $p : p = \hat{p} \pm z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}}$. $\hat{p}$ is the correct prediction proportion of the sample, and n is the sample size. $z_{\alpha/2}$ is the confidence coefficient, and it equals to 2.576 when the confidence interval is 99%. The result is

$$p = \frac{\hat{p} + \frac{z^2_{\alpha/2}}{2n} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2_{\alpha/2}}{4n^2}}}{1 + \frac{z^2_{\alpha/2}}{n}}. \tag{7}$$

Then, the upper bound of confidence interval of global prediction under the new method is

$$p_{global} = \frac{p'_{global} + \frac{z^2_{\alpha/2}}{2(n-1)} + z_{\alpha/2}\sqrt{\frac{p'_{global}(1-p'_{global})}{n-1} + \frac{z^2_{\alpha/2}}{4(n-1)^2}}}{1 + \frac{z^2_{\alpha/2}}{n-1}}. \tag{8}$$

From the result, according to knowledge of the infinitesimal of higher order of the limit theory, we can see that when $n \to \infty$, the equation is approximate to

$$p_{global} = p'_{global} + z_{\alpha/2}\sqrt{\frac{p'_{global}(1-p'_{global})}{n-1}}. \tag{9}$$

This means that the formula is also applicable to the case of large sample datasets. Besides, when $p'_{global} = 0$, the result is greater than 0, which indicates that it can also handle the situation of endpoint values.

Last but not least, we retain the original local prediction during the process of estimation because it is valid regardless of the sample size and extreme probability. Then, we choose the maximum of the global and local prediction to calculate the min-entropy as the final result.

4.5 Setting of Key Parameters

In this section, we discuss the setting of the parameters. We choose an initial accumulation size of 5000 samples for dictionaries and scoreboards due to the fact that the largest sliding window of the MultiMCW prediction estimator is 4095. If the accumulation size is smaller than 4095, the largest sliding window cannot accumulate dictionaries and scoreboards for the initial samples. This setting is a conservative approach, and it is suitable for other prediction estimators in 90B.

When determining the size of the interval I in Figure 1, we take into account it both from theoretical and experimental perspectives. On the one hand, as mentioned earlier, the MultiMCW estimator's largest sliding window has a size of 4095. Thus, the interval I should be greater than this value, and we also prove it through the experiment. On the other hand, we conducted an experiment to determine the upper bound. We set the intervals as 2^k, and k takes from 1 to 17, and evaluated sequences that followed IID and non-IID distributions separately. Because the dataset within each segment is stationary after segmentation under change detection technology, we needn't use the time-varying sequence here. For the IID dataset, we select a typical dataset generated by the Oscillator-based model [1]. For the non-IID dataset, we choose one that followed a Markov

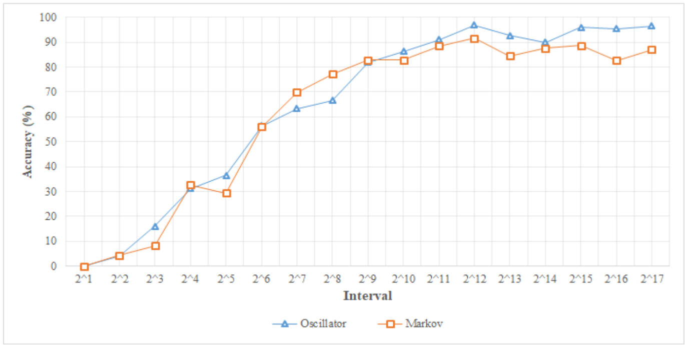


Fig. 2. Accuracy under Different Intervals.

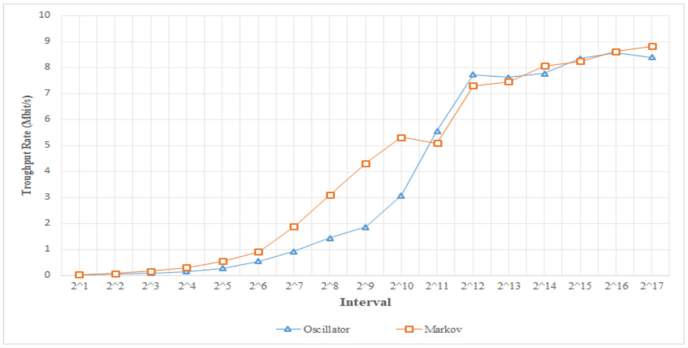


Fig. 3. Throughput Rate under Different Intervals.

model. The accuracy and throughput rate under different intervals are depicted in Figure 2 and Figure 3.

In the results, we use the line chart to depict the accuracy and throughput rate. We see that the accuracy improves as the interval size increases, and when the interval exceeds 2^{12}, the accuracy starts to fluctuate around 90%. Besides, with larger intervals, the throughput rate grows faster, and when the interval reaches 2^{12}, the throughput rate gradually becomes stable. However, processing too much data at once may consume a significant amount of memory and lead to latency. Therefore, to ensure accuracy and throughput rate, we set the interval range from 2^{12} to 2^{17}. For the sake of convenience in displaying the results throughout the rest of this paper, we set a fixed interval of 50000.

5 Experiment Results and Analysis

5.1 Experiment Setup

Our estimator is implemented in C/C++ language, and we show the results of the other estimators for comparison. In this section, all experiments are conducted on a Windows 11 system with an Intel 11th Gen Intel(R) Core(TM) i7-1195G7 CPU and 16GB of memory.

During the experiment, we present the results in two ways. Firstly, for the offline estimators such as the estimators in 90B and others based on deep learning algorithms [10, 12, 21], we only compare their final offline estimation results with the endpoint result of our proposed estimator. We then display the error rate in the figure. This is because their methods are exclusively used in offline scenarios, and it would be unreasonable to choose intermediate output results for the final comparison. The error rate is calculated by the following formula:

$$ErrorRate = \frac{|H_{test} - H_{correct}|}{H_{correct}} \times 100\%, \tag{10}$$

where $H_{correct}$ is the theoretical min-entropy, and H_{test} is the results of the estimators.

Secondly, for the online estimators, including our proposed one and the online estimator based on collision entropy proposed by Kim [9], we plot their estimations in the figures. We do not present the values of the 90B estimators implemented on FPGA because they utilize outdated estimators of the old version of 90B standard, which have some mistakes [6, 19].

5.2 Simulated Datasets for Experiments

The datasets used in our experiment can be divided into two categories: stationary datasets and time-varying datasets. The stationary datasets comprise various distribution families, including discrete uniform distribution, discrete near-uniform distribution, and normal distribution rounded to integers. More details are provided below.

– **Discrete Uniform Distribution:** The samples are subject to the discrete uniform distribution and are equally-likely. They come from an IID source.

– **Discrete Near-uniform Distribution:** The samples are subject to the discrete near-uniform distribution with one higher probability than the rest. They come from an IID source.

– **Normal Distribution Rounded To Integers:** The samples are subject to normal distribution and are rounded to integer values. They come from an IID source.

The time-varying datasets consist of two common situations: mutation (i.e., sudden change) and gradient (i.e., gradual change). To represent mutation, we utilize a dataset that undergoes near-uniform distribution with 9 mutations. For the gradient scenario, we employ a Markov model that exhibits a gradient following a linear function curve. The specific details are outlined below.

– **Discrete Near-uniform Distribution with Mutation:** The samples are divided into ten parts and each subject to the discrete near-uniform distribution with different parameter values, i.e., the higher probability. Table 2 shows the changes.

– **Markov Model with Gradient:** The samples are subject to a first-order Markov process of $\{0, 1\}$, and its transfer matrix is $\begin{pmatrix} 1-p & p \\ p & 1-p \end{pmatrix}$, where p changes along a linear function curve:

$$p(i) = \begin{cases} 0.1 + 0.0000004i & , 0 \leq i < 500000 \\ 0.3 & , 500000 \leq i < 1000000 \end{cases}, \tag{11}$$

where i is the serial number of the sample.

Table 2. Discrete Near-uniform Distribution with Mutation.

Serial Number of the Sample	**Higher probability**
[1, 80000]	0.5
[80001, 230000]	0.8
[230001, 330000]	0.6
[330001, 380000]	0.85
[380001, 400000]	0.7
[400001, 600000]	0.9
[600001, 900000]	0.55
[900001, 1200000]	0.75
[1200001, 1350000]	0.95
[1350001, 1500000]	0.65

In terms of the dataset size, our proposed online estimator only requires a minimum of 2^{12} samples. However, other offline estimators, as per the 90B standard, necessitate no less than one million samples. To facilitate comparison, we utilize a dataset that follows a discrete near-uniform distribution with 1.5 million samples for the mutation scenario, and other datasets with 1 million samples for the remaining scenarios.

5.3 Experimental Results

In this subsection, we present the results and then analyze them.

1) Offline estimation results

In figure 4, we use a column chart to represent the comparison of the error rate of ours and other estimators under different sequences. We find that our online estimator can give better results than raw 90B estimators and other estimators that use deep learning algorithms, especially in time-varying sequences.

2) On-the-fly test results

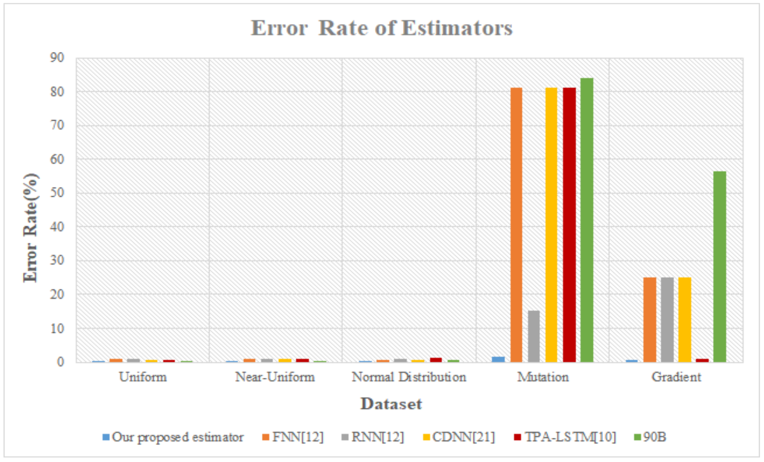


Fig. 4. Comparison of error rate of estimators.

In the figures, we denote the correct values with red line, and use green "+" dots representing the results of the online estimator based on collision entropy. Ours is shown by blue "x" dots.

Figure 5(c) shows the estimated results of the simulation dataset from the independent normal distribution entropy source. We observed that the results provided by our estimator and the online estimator based on collision entropy are both close to the correct entropy.

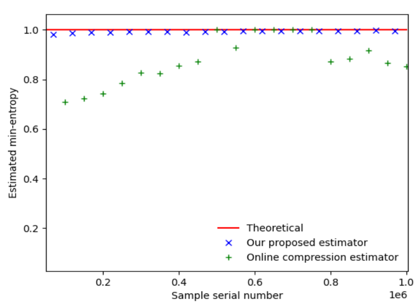
Figure 5(a) and Figure 5(b) shows that the online estimator based on collision entropy always provide severely underestimation results on the datasets subject to discrete uniform distribution and discrete near uniform distribution because its algorithm is too simple to mine out the features of the datasets. By contrast, ours provides almost accurate estimations.

For the time-varying datasets, the online estimator based on collision entropy completely deviates from the theoretical min-entropy in Figure 6(a) and Figure 6(b). The estimations of ours can approach the theoretical correct values at most points due to the timely clearing of the dictionaries, although there are some deviations at the inflection points. This is caused by the delay in the change detection technology, and the delay is quite small, which is less than 1000 samples. It proves the effectiveness of our change detection technology.

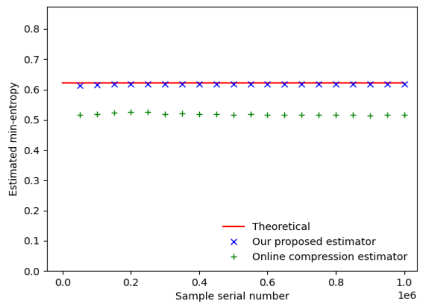
5.4 Performance Evaluation

In this section, we discuss the performance of our proposed estimator. Firstly, in above results, our estimator can give more accurate estimation than other estimators.

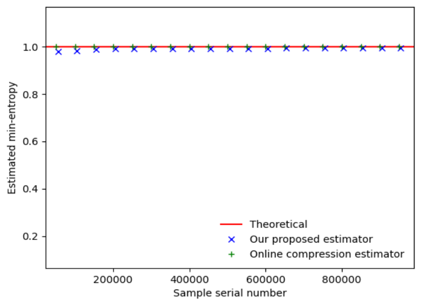
Secondly, in terms of time consumption, as is shown in Table 3, the throughput rate of our estimators is stationary under different datasets, which is about 8.85 Mbit/s. Therefore, for on-the-fly test, our estimator is suitable for random



(a) Comparison of min-entropy estimators for uniform source.



(b) Comparison of min-entropy estimators for near-uniform source.



(c) Comparison of min-entropy estimators for normal distribution source.

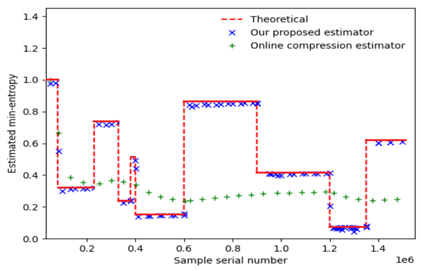
Fig. 5. Comparison of min-entropy for stationary sequences.

number generators with throughput rates less than or equal to 8.7 Mbit/s in terms of conservative estimation, whether software or hardware random number generators.

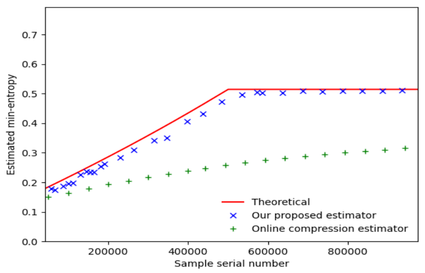
Besides, in the real world, entropy is an issue on low-power devices. Our estimator consumes 300 Mbit of memory for processing 1Mbit of data, which is the same as the raw 90B standard.

6 Conclusion

In this paper, we design a new estimator based on the 90B prediction estimators for on-the-fly test. This design enhances both speed and accuracy. By employing change detection technology in our proposed new framework, we have achieved excellent performance. Additionally, to address situations involving small sample datasets or extreme probability, we utilize the "Plus Four Confidence Intervals" method to calculate the global predictability. Our estimator achieves a throughput rate exceeding 8.7 Mbit/s, meeting the on-the-fly test requirements of many RNGs. It currently stands as the most accurate technology for evaluating min-entropy. Looking ahead, our future plans involve further improving



(a) Comparison of min-entropy estimators for near uniform mutation sources.



(b) Comparison of min-entropy estimators for Markov model gradient source.

Fig. 6. Comparison of min-entropy for time-varying sequences.

Table 3. Throughput rate under different sequences.

Data type	Throughput rate (Mbit/s)
Uniform	8.92
Near-uniform	8.90
Normal distribution	8.85
Mutation	8.76
Gradient	8.82

speed through hardware enhancements and parallel computing, aiming to ensure compatibility with a broader range of entropy sources.

Acknowledgements This work was supported by the National Natural Science Foundation of China under Grant 62272457.

References

1. Baudet, M., Lubicz, D., Micolod, J., Tassiaux, A.: On the security of oscillator-based random number generators. Journal of cryptology **24**(2), 398–425 (2011)
2. Box, J.F.: Gosset, fisher, and the t distribution. The American Statistician **35**(2), 61–66 (1981)
3. Clopper, C.J., Pearson, E.S.: The use of confidence or fiducial limits illustrated in the case of the binomial. Biometrika **26**(4), 404–413 (1934)
4. Feller, W.: An introduction to probability theory and its applications, Volume 2, vol. 81. John Wiley & Sons (1991)
5. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17. pp. 286–295. Springer (2004)
6. Grujić, M., Rožić, V., Yang, B., Verbauwhede, I.: Lightweight prediction-based tests for on-line min-entropy estimation. IEEE Embedded Systems Letters **9**(2), 45–48 (2017)

7. Kelsey, J., McKay, K.A., Sönmez Turan, M.: Predictive models for min-entropy estimation. In: Cryptographic Hardware and Embedded Systems–CHES 2015: 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings 17. pp. 373–392. Springer (2015)
8. Killmann, W., Schindler, W.: A design for a physical rng with robust entropy estimators. In: Cryptographic Hardware and Embedded Systems–CHES 2008: 10th International Workshop, Washington, DC, USA, August 10-13, 2008. Proceedings 10. pp. 146–163. Springer (2008)
9. Kim, Y., Guyot, C., Kim, Y.S.: On the efficient estimation of min-entropy. IEEE Transactions on Information Forensics and Security **16**, 3013–3025 (2021)
10. Li, H., Zhang, J., Li, Z., Liu, J., Wang, Y.: Improvement of min-entropy evaluation based on pruning and quantized deep neural network. IEEE Transactions on Information Forensics and Security **18**, 1410–1420 (2023)
11. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE transactions on knowledge and data engineering **31**(12), 2346–2363 (2018)
12. Lv, N., Chen, T., Zhu, S., Yang, J., Ma, Y., Jing, J., Lin, J.: High-efficiency min-entropy estimation based on neural network for random number generators. Security and Communication Networks **2020**, 1–18 (2020)
13. Ruk, A., et al.: A statistical test suite for the validation of random number generators and pseudo-random number generators for cryptographic applications. NIST, http://csrc. nist. gov/rng/rng2. html (2001)
14. Santoro, R., Tisserand, A., Sentieys, O., Roy, S.: Arithmetic operators for on-the-fly evaluation of trngs. In: Mathematics for Signal and Information Processing. vol. 7444, pp. 253–264. SPIE (2009)
15. Turan, M.S., Barker, E., Kelsey, J., McKay, K.A., Baish, M.L., Boyle, M., et al.: Recommendation for the entropy sources used for random bit generation. NIST Special Publication **800**(90B), 102 (2018)
16. Veljković, F., Rožić, V., Verbauwhede, I.: Low-cost implementations of on-the-fly tests for random number generators. In: 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 959–964. IEEE (2012)
17. Wilson, E.B.: Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association **22**(158), 209–212 (1927)
18. Yang, B., Rožić, V., Mentens, N., Dehaene, W., Verbauwhede, I.: Embedded hw/sw platform for on-the-fly testing of true random number generators. In: 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 345–350. IEEE (2015)
19. Yang, B., Rožić, V., Mentens, N., Verbauwhede, I.: On-the-fly tests for non-ideal true random number generators. In: 2015 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 2017–2020. IEEE (2015)
20. Yang, J., Zhu, S., Chen, T., Ma, Y., Lv, N., Lin, J.: Neural network based min-entropy estimation for random number generators. In: International Conference on Security and Privacy in Communication Systems. pp. 231–250. Springer (2018)
21. Zhu, S., Ma, Y., Li, X., Yang, J., Lin, J., Jing, J.: On the analysis and improvement of min-entropy estimation on time-varying data. IEEE Transactions on Information Forensics and Security **15**, 1696–1708 (2019)