An Exploration of Machine Learning Cryptanalysis of a Quantum Random Number Generator

Abenezer Monjor
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
monjo003@morris.umn.edu

ABSTRACT

Random number generators (RNG) are important for statistical sampling, computer simulation, cryptography, and other applications where producing unpredictable results is crucial. Due to the deterministic nature of these RNG, applications that depend upon unpredictability, such us cryptography may be exposed to adversarial attack. Quantum Random Number Generators (QRNG) are inherently random because of the quantum mechanics core, which makes quantum systems a great source of entropy. In this paper, we will walk through how by using a predictive machine learning analysis, we can see the influence of deterministic noise in multiple stages of an optical continuous QRNG. The machine learning model used in the investigation detects inherent correlation if the deterministic sources are prominent.

Keywords

Cryptography, neural network, activation function, adversarial attack

1. INTRODUCTION

There have been numerous stories about NSA putting a backdoor in RSA cryptography back in 2014. Which would allow the NSA to determine the output of the random number generator used in the RSA protocol. According to Schneier[4] breaking random-number is basically like breaking the entire security system because badly designed slow algorithms gives a backdoor for an attack. Due to that there is a need to have pretty good secured random number generators to secure computer systems.

Random number have been used in many applications. For example, they were used in cipher encryption which was used during the second war, and now a days, they are used to securely browse the internet, and for many other applications in cryptography (eg. Smart Energy Grid, e-banking, internet trade, prepaid cards etc...). They are also used outside of cryptography in fields like statistical sampling, simulation, gaming, and gambling.

There are different ways of generating random numbers [5], two well known approaches to generate random numbers are by using a software algorithm, also known as Pseudo-random number generators (PRNG) and by using Hardware random number generators (HRNG). PRNG is a mathematical formula that produces a sequence of numbers that are determined by the initial input known as the seed. In contrast HRNG generate sequences of numbers from a non-deterministic physical process, which makes them truly random.

One type of HRNG is quantum random number generators (QRNG), which is the main subject of this paper. We are going to explore a paper written by Nhan Duy Truong, Jing Yan Haw, Syed Muhamad Assad, Ping Koy Lam, and Omaid Kavehei titled Machine Learning Cryptanalysis of a Quantum Random Number Generator [6]. In their paper they use predictive machine learning in different stages of an optical QRNG to investigate the impacts of deterministic classical noise. They use a machine learning model to recognize patterns in a given sequence of randomly generated numbers to investigate how classical entropy affects the unpredictability and randomness of a vacuum fluctuation based QRNG(see Haw, et all's article [2] for details). They applied the machine learning model in two situations. The first only measured the classical entropy coming out of the QRNG, and the second measured the combined effect of the quantum and classical noise, to investigate how much the classical entropy affects the QRNG's unpredictability and randomness.

2. BACKGROUND

To understand the paper [6] there are some key concepts that need to be elaborated. These concepts include randomness, random number generators, correlation, adversarial attack, and neural networks.

2.1 Randomness

Randomness can be defined as a lack of predictability in a sequence of events. Some times things are unpredictable because we don't understand the outcome of physical properties (eg. rolling a dice), but some times randomness can exist because some things are truly random. The type of randomness that is the result of complexity and our ignorance of the system is called classical randomness which is deterministic, whereas the result of randomness caused by quantum mechanics is called quantum randomness, which is non-deterministic.

For instance, soccer referees flip a coin for the teams to decide which goal they will attack in the first half of the match, and it is believed that flipping coin is something random, but it is not truly random. Because if we would have known how the coin started to flip, the forces acting

UMM CSci Senior Seminar Conference, November 2019 Morris, MN.

on it, and if we are brave enough to calculate that, we could have predicted the outcome. Therefore we can say flipping a coin is random because of the hidden variables that we didn't take into account. This can be an example of classical randomness, which is deterministic. An example of a non-deterministic type of randomness is a randomness extracted from radioactive decay, which is a process of unstable atomic nucleus losing energy. The time frame of a particular atom decay is completely random and it is impossible to predict when a single atom will decay.

2.2 Entropy

Entropy is a measurement of a system's disorder. In statistics entropy can be defined as a logarithm of different ways of arrangements a system can be configured. It can be described as follows:

$$S = -\sum_{i}^{n} P_i \ln P_i$$

where P_i is the probability of each outcome. Entropy has nothing to do with the actual values of an outcome, but it measures the relative probabilities of the outcomes. Therefore it is maximum when all the outcomes are equally likely and it is always greater than or equals to zero.

2.3 Random Numbers Generators

Random number generators (RNG) are devices that generate a sequence of a random number that can not be predicted better than a guess. There are generally two kinds of RNG: Hardware random number generators (HRNG) and pseudo-random number generators (PRNG). HRNG are non-deterministic, meaning events on the same input can result different behavior in different runs, which makes them to be more secure than PRNG. Whereas PRNG are deterministic, which means given a particular input always results in the same output. One good example of HRNG are quantum random number generators (QRNG).

2.3.1 Pseudo-random Number Generator

Pseudo-random numbers generator is a deterministic algorithm to generate a sequence of numbers that have an approximate property of random numbers. Pseudo-random number generators are not truly random, but their randomness is enough for some applications. When sequences are generated there are many sequences that can not occur. Although they are not perfectly secure, they are *practically* secure and used for some applications in cryptography and simulation.

Generating pseudo-random numbers start by inputting a seed, which can be generated from a current-time in milliseconds or from a measurement of noise, into a deterministic algorithm. After inputting the seed, the algorithm uses the output of the seed as the next seed and repeats the process as many times as needed. The deterministic algorithm [3], for example, can be based on number theory (i.e linear congruential generators, they produce random numbers from a recursive formula). See article [1] for more details.

$$x_{n+1} = (ax_n + c) \mod m, n > 0$$

where x_i is the ith digit in the random number sequence, M > 0 is the modulus, 0 ≤ c < m is the increment and 0 ≤ a < m is the multiplier.

PRNG can also be generated from a linear shift feedback registers (LSFRs), by using right shift operation and an XOR operation, A basic LFSR doesn't produce very good random numbers [3]. To have a better random number, a larger LFSR with a lower bit is should to be used.

There are many advantages of using PRNG over hardware random number generators. Some of the reasons are PRNG are much faster than true random number generators, and same sequence of generated random numbers can be repeated upon demand, which is very helpful to look for errors in calculation that rely on random numbers.

2.3.2 Quantum Random Number Generator

Quantum random number generators (QRNG) are devices that uses quantum random properties to generate true random numbers. As opposed to deterministic RNG, where randomness is the result of uncertainty [3].

QRNG are very important in cryptography, but they also have some limitations. Some of their limitations are they are slow compared to PRNG, adding an external device is not usually convenient, and it is hard to detect if there are failures or errors.

The type of QRNG that is used in the paper [6] is an optical quantum random number generator that has two segments: entropy source and post-processing procedure [6]. The entropy source produces raw randomness as a result of a physical process. This raw randomness contains both classical and quantum noises [6]. The noise that comes from the classical entropy is from devices such as analog to analog converters and peripheral measuring devices. To produce pure randomness the classical entropy has to be removed from the main random bit stream since it is untrusted [6]. The post-processing block is what is used to extract pure quantum randomness from the device.

2.4 Correlation and Auto-correlation

Correlation is a statical association between two random variables, which is used to indicate a predictive relationship that can be used in practice.

Auto-correlation measures similarity between a variable's current value in a given time-serious with its lagged version over-successive time interval. A time-series is a measurement of the same variable over time. Auto-correlation is useful to detect if a given sequence of numbers are not random and it can be expressed as the following formula, where $Y_1, Y_2, ..., Y_N$ are the given measurements. $X_1, X_2, ..., X_N$ is the given time period.

$$r_k = \frac{\sum_{i=1}^{N-k}(Y_i - \bar{Y})(Y_{i+k} - \bar{Y})}{\sum_{i=1}^{N}(Y_i - \bar{Y})^2}$$

2.5 Neural Network

To find a correlation between the generated random numbers by using only classical noise versus using both the quantum and classical noise, neural network can be a great tool. Neural network are inspired by the structure of the human brain. As our brain has neurons that are connected with each other by synapses, Neural networks have nodes that are connected to edges and each edge has a weight based on its influence on another node.

Neural network have multiple layers. Between the first and last layers, there could be different layers, which are called hidden layers: each of them distilling a particular feature.

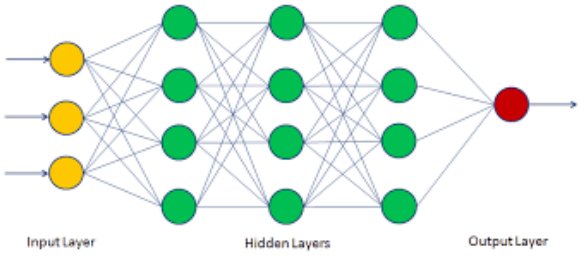

Figure 1: Example of neural network containing three hidden layers.

Figure 1 shows a neural network model that has three hidden layers. Each of the nodes in a layer updates their values by taking a weighted sum of the values of the nodes in the previous layer. The weights are assorted to the connection between the layers shown as lines on figure 1. Training a neural network amounts to finding good values for those weights (see section 2.6). A weight can either dampen or amplify the influence of one node upon another.

On the paper [6] a neural network is used to forecast what the next output bit of the QRNG is going to be. The class of neural network used in [6] is called recurrent convolutional neural network and it is a combination of recurrent neural network and convolutional neural network.

2.5.1 *Convolutional Neural Network*

Convolutional Neural Network (CNN) are one class of feed-forward neural networks. Feed-forward neural network are a simple type of neural network where information moves only in the forward direction. Which means information moves forward from the input and it goes to the hidden layers, and after a series of calculations of events it results in output.

CNN are well known for their use in analyzing visual imagery [6]. They were inspired by the biological process of the visual cortex. As individual neurons in the cortex respond only if there are stimuli in the receptive field, each neuron in one layer goes to the next layer only if it receive an input from the previous layer. Most of the time CNNs are composed of sets of layers that can be grouped by their functionalities called convolutional layers.

2.5.2 *Recurrent Neural Network*

Recurrent Neural Networks (RNN) are a powerful class of neural networks that can be used to find potential patterns of a long sequences of generated random numbers. They generate outputs using both the current input and values from the previous computations. Which means the output of the results are used as an input for the current step.

RNNs have been very successful in speech recognition and language modeling [6]. They work similarly to our brain works when we watch a movie. For example if we want to classify what is going to happen at every point of a movie, we have to go back and understand what happened in the previous scenes, because we can't make a good guess of what is going to happen just by watching one scene of a movie. RNNs addresses this same issue.

Long-short term memory (LSTM) is one of RNN's architectures, which is capable of learning long-term dependencies. In the neural net that is used to make predication in the QRNG, there may be hidden long term dependencies that are not obvious to human brains, and their goal is to detect those dependencies, in order to increase the ability of RCNN to make good predication [6].

2.5.3 *Recurrent convolutional neural network*

As its name implies recurrent convolutional neural networks (RCNN) are the combination of recurrent neural network and convolutional neural network. [6] RCNN are implemented by feeding features extracted by convolutional layers into RNN [6]. The neural net used in [6] has two convolutional layers and one LSTM (see fig 3). The convolutional layers look for features in the input, and the LSTM detect longer and shorter term dependencies, together the architecture is designed as effective as possible to predict the next term in a random sequence, by looking at previous values.

2.6 Model Training

In the paper we are analyzing the QRNG generates a sequence of integers. The neural network is given 100 of those integers and tries to predict the next one. Before the neural network can make predictions it is trained. Training consists of using test data to iteratively improve the performance of the neural network.

The weights associated to the edges in a neural network are used in the calculation that the neural net performs to make its predictions. If the weights are adjusted then the prediction can change. There is a technique (which we won't cover in this paper) for updating the weights. Each data-point in the training set must have a target value known as a label. A data-point is provided to the neural network and the prediction is compared to the label. The differences between the prediction and the label are then used to adjust the weights in a way that makes the prediction a little bit more likely to be more accurate. This is repeated for thousands or millions of times until, hopefully, the neural network becomes good at making predictions.

color	color_red	color_blue	color_green
red	1	0	0
green	0	0	1
blue	0	1	0
red	1	0	0

Figure 2: Example of one hot encoding.

2.6.1 *One Hot Encoding*

One hot encoding is process of converting categorical variable into a form that could be provided to the model as an input. For example in fig 2 we can classify the colors red, green, and blue, which are categorical data and convert them into numeric vectors where red represents the vector [1, 0, 0], green [0, 1, 0], and blue [0, 0, 1].

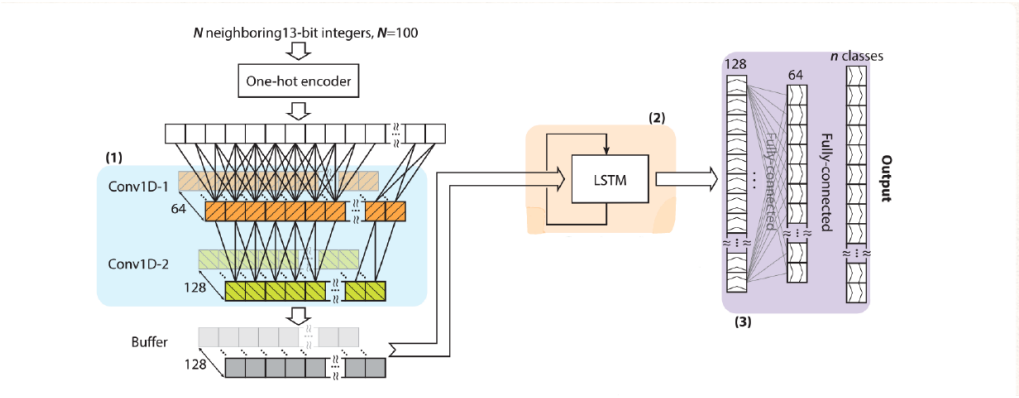

Figure 3: Figure modified from [6] recurrent convolutional neural network (RCNN) model: The 100 one hot encoded 13 bit integers goes to a serious of convolutional kennels, the output is then generated by a buffer and goes to the LSTM, the output of that goes to two fully-connected layers that output one single one-hot encoded 13 bits integer

3. PAPER DETAIL

The main objective of Duy Truong, et all's paper is to use a neural network model to predict what the next generated random number is going to be when there is a quantum source versus when there is only classical noise. The neural network used for this purpose is RCNN, and it is used to find potential patterns in the generated random numbers at the different stages of the QRNG. The model is trained with N data value to guess what the next generated number is. The proportion of accurate output values of the model is denoted by P_{ml} whereas the probability of what the most likely out is denoted by P_g, which is a baseline strategy for making guess by observing the most common number to be put out by the QRNG. The P_{ml} is compared against P_g.

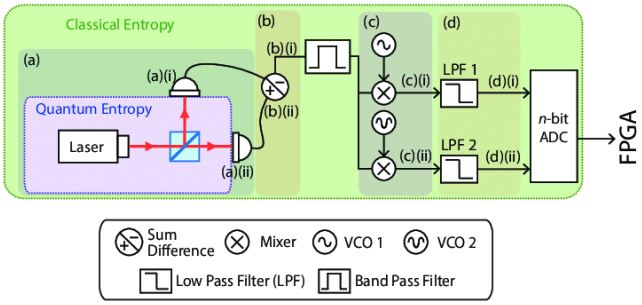

Figure 4: Data acquisition stages in the entropy blocks of the QRNG. Stage (a): (i) detector 1 and (ii) detector 2; Stage (b): (i) difference and (ii) sum of the photo-currents; Stage (c): difference of the photo-currents demodulated Stage (d): Low pass filtering of the signals from (c). Picture taken from [6].

3.1 Data acquisition stages of the QRNG

The QRNG has two main parts. The first part is an entropy source, which produces raw randomness that contains both classical and quantum noise. The second part is post-processing procedure, where the pure quantum randomness is extracted from the combined classical and quantum entropy.

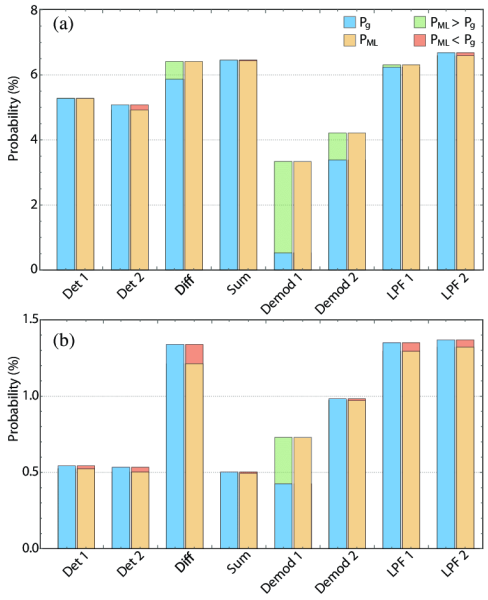

Figure 5: Prediction performance of the deep learning model. (a) P_{ml} surpasses P_g in 4 out of 8 stages (Diff, Demod 1, Demod 2 and LPF 1) in scenario 1 (classical) by more than 2 standard deviations (not shown on the plots) and (b) only 1 out of 8 stages (Demod 1) in scenario two (quantum and classical). Fig taken from [6]

To see how much the classical noise affects the QRNG, data is collected in different stages of the QRNG. As indicated on Figure 4 on the first stage (Figure 4 a), there is a laser beam that is projected on a half silvered mirror. Which results in the light beam going vertical half of the time and horizontal half of the time. The photo detectors (Fig. 4(a)(i) and a(ii)) detects the light intensity of the photons and convert them in to electric current. In stage b the currents are combined together (which will eventually result in the random values generated by the machine). Then the wave gets passed through a band pass filter, which only lets

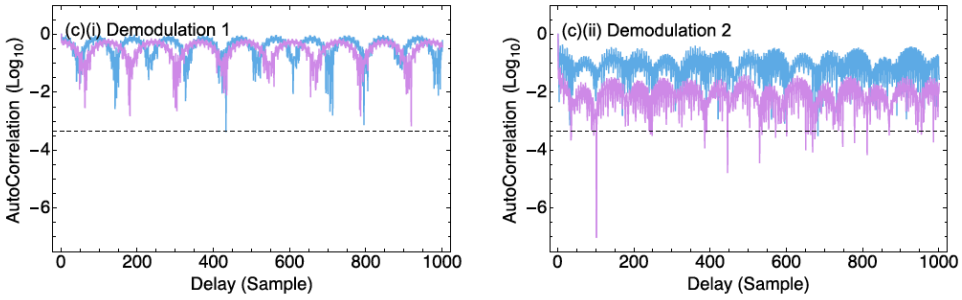

Figure 6: A sample of stage taken from [6] to show auto-correlation of data-sets in scenario 1 (blue), where only electrical noise is present and scenario 2 (pink), where both electrical and quantum noise are present. Dashed lines show the theoretical standard deviation of truly random 5 million samples.

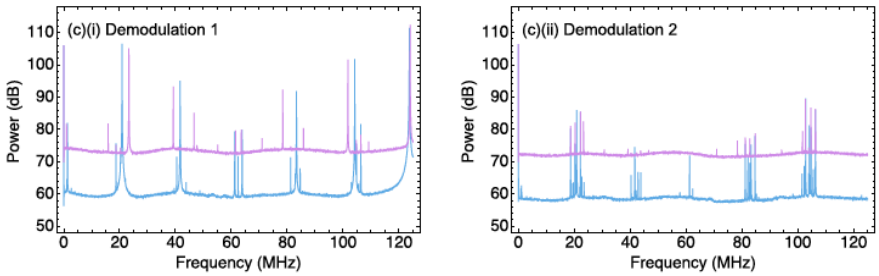

Figure 7: Sample of stage taken from [6] to show power spectrum of datasets in scenario 1 (blue), where only electrical noise is present and scenario 2 (pink), where both electrical and quantum noise are present.

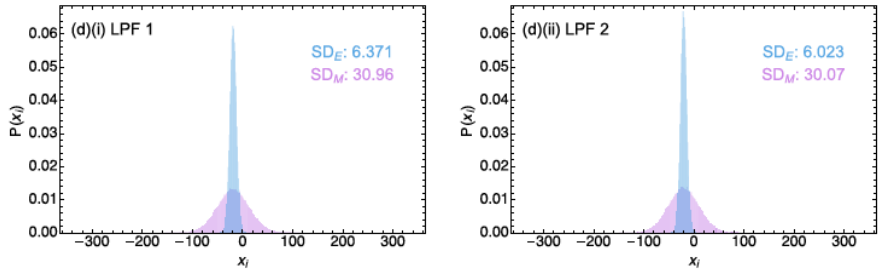

Figure 8: Sample of a stage taken from [6] shows the different in probability distribution when there is only classical noise (blue) versus when there is both classical and quantum noise (pink) in the QRNG.

frequencies between two cutoff to pass through. On stage (c) those cutoffs are demodulated at 1.375 GHz and 1.625 GHz. In stage (d) the signal is passed through a low-pass filter to cut off the high frequencies. Those signals goes to an analogue digital converter which produces a single 16 bit integer. The 16 bit integers are latter converted into a 13 bit integer, because one hot encoding a 13 bits integer in much easier than one hot encoding a 16 bit integer.

3.2 Data set and model training

Ten million 13-bit integers are used in the model. The first 5 million are used to train the model. The remaining 5 million integers are used in 5 test-sets.

The data is 10 million 13-bit integers considered as a sequence. 101 of those integers are used as a single "data point": (X_1, ..., X_{101}) Think of those 100 values as being determined by a sliding window that is advanced by 3 steps. The next data point is also therefore (X_4, ... , X_{104}), and so on.

For any given data point, the first 100 integers are one-hot encoded and act as the input to the neural net. The remaining integer is considered to be the target (or label) that the neural net is trying to predict. The convolutional layers attempt to find local features that will help in the prediction and the LSTM layers are included to help detect longer-term correlations.

The 100 13-bit integers are encoded as one hot vectors (see section 2.6.1). As it is shown on Figure 3 the one hot vector goes to two convolutional layers. The output of the second convolutional layer is stored in a buffer in order to the subsequent LSTM layer to have access to outputs from previous data points. The output the LSTM is connected to 2 fully connected layers. The two layers output a one hot encoded single 13 bit integer value.

3.3 How the system is evaluated

When P_g is compared with P_{ml}, the P_g corresponds to the highest probability of a single integer in the probability distribution of the integers that are extracted from the QRNG. In figure 3 we see an example from the LPF1 stage, the most common integer is -26 as can be seen from the location of the peak. The value has the highest success rate (1.37%). This value is compared with P_{ml} in section 4 and figure 9 in particular. If there happens to be correlation between the generated numbers P_{ml} will give a better accuracy result compared to P_g.

4. RESULTS

In the first scenario, where only the classical noise is considered, the RCNN model demonstrated its ability to find correlations between the generated numbers. Whereas in the second scenario, where both the classical and quantum noises considered, the RCNN model shows a considerable decrease in predicting what the next generated random number is [6]. This can be seen in figure 9, Where the x-axis showing the different stages of the QRNG, and the y-axis shows the probability of success. As mentioned we will be comparing P_g and P_{ml} and discuses in details below.

4.1 Classical Entropy

In the first experiment, only a classical source of entropy was used. The RCNN model shows P_{ml} was better in 4 out of 8 stages than the P_g in guessing what the next generated random number is going to be [6]. As it is shown on the diagram 5 fig a, we can see P_{ml} is greater than P_g in the stages Diff, Demod 1, Demond 2 and LPF 1.

Fig 6 - 9 show the probability distribution, the auto correlation, and power spectral density (PSD) fot the different stages of the QRNG for both scenarios (purely classical vs both classical and quantum). The power spectral density graph shows the strengths of the frequencies in a signal. As an analogy, if Figure 7 were showing the PSD for a piece of music, then the peaks would indicate the loudest tones (frequencies) in the score.

For the measurements in both scenarios, the blue color shows when there is only classical noise, and the pink color indicates when there is both classical and quantum noise. Since we can't explain all the probability distribution and autocorrelation for each steps in the QRNG in this paper, we look at only a few stages. We can see from the sample probability distribution diagram Fig 8, as seen when there is only classical noise there is a pretty tight distribution, whereas when there is both classical and quantum entropy we see a wide probability distribution meaning there are more possible outcomes that can be generated when there is both quantum and classical noise than when there is only a classical noise in the system.

Also in stage b (corresponds to Diff on Fig 5), where they have the difference of the detectors, we can see the RCNN model is better than P_g, in capturing patterns and predicting the next outcome. Again in stage c(corresponds to Demod 1 on Fig 5), P_{ml} makes great prediction which is almost six time better than the P_g. The auto-correlation (Fig 6) plots show repetitiveness, due to that the P_{ml} is able to make a really good predication.

4.2 Classical and Quantum Entropy

When both quantum and classical noises were considered, there was a huge decrease in the ability of the RCNN model to determine what the next random number would be. Out of the 8 different stages in only one of them, the RCNN model was better at predicting what the next outcome would be. As it can be seen from the Fig 5 b only in one of the 8 stages (on stage Demod 1) the P_{ml} is better at determining the next outcome than P_g.

As we can also see from the results of the auto-correlation Fig 6, there is not much of noticeable repetitiveness when there is both classical and quantum noise in the system. Another way of looking at the result is by looking at the PSD Fig 8, where again we see a cyclic repetitiveness when there is only classical noise whereas when there is both classical and quantum noise in the system, there is a decrease in the cyclic repetitiveness. Due to the decrease in the autocorrelation, PSD, and probability distribution, the P_{ml} was not able to make good prediction when there is both classical and quantum noise.

5. CONCLUSIONS

By applying a machine learning algorithm on [6], Duy Truong, et all's were able to find very little pattern in the various stages of the QRNG. In only one situation was the machine learning prediction, P_{ml} substantially better than just guessing the most commonly observed integer, P_g

Acknowledgments

I would like to thank Peter Dolan and KK Lamberty for their advice and feedback.

6. REFERENCES

[1] A. Belsare, S. Liu, and S. Khatri. Gpu implementation of a scalable non-linear congruential generator for cryptography applications. In *Proceedings of the 23rd ACM International Conference on Great Lakes Symposium on VLSI*, GLSVLSI '13, pages 89–94, New York, NY, USA, 2013. ACM.

[2] J. Haw, S. Assad, A. Lance, N. Ng, V. Sharma, P. Lam, and T. Symul. Maximization of extractable randomness in a quantum random-number generator. *Physical Review Applied*, 3(5), May 2015.

[3] M. Herrero-Collantes and J. C. Garcia-Escartin. Quantum random number generators. *Reviews of Modern Physics*, 89(1), Feb 2017.

[4] B. Schneier. Schneier on security, Nov 2007.

[5] M. Stipcevic. Quantum random number generators and their applications in cryptography. *Advanced Photon Counting Techniques VI*, May 2012.

[6] N. D. Truong, J. Y. Haw, S. M. Assad, P. K. Lam, and O. Kavehei. Machine learning cryptanalysis of a quantum random number generator. *IEEE Transactions on Information Forensics and Security*, 14(2):403–414, Feb 2019.