Multiple-Valued Random Digit Extraction

Micah A. Thornton
Department of Statistical Science
Southern Methodist University
Dallas, Texas, USA
mathornton@smu.edu

Mitchell A. Thornton
Darwin Deason Institute of Cybersecurity
Southern Methodist University
Dallas, Texas, USA
mitch@smu.edu

***Abstract*—We describe a parameterized random value extraction function for producing symbol strings based upon a user-specified radix value. The presentation of the extractor function is accompanied with a theoretical discussion that describes how the radix value, and hence the symbol set cardinality, affects the entropy of an extracted symbol string. Experimental results are provided where extracted strings with respect to different radix values are analyzed for randomness quality. The empirical data is obtained by applying our parameterized extractor function to a weakly random source that provides a set of inter-packet delay spacings observed from a network interface card in a general-purpose computer. Our results show how the quality of the extracted random strings varies for differing radix values. These results provide guidelines for choosing an appropriate radix value in an extractor function.**

I. Introduction

We describe the formulation and use of a multiple-valued entropy extractor function where the radix value is a parameter of the extractor. The parameterized radix value allows the extractor to produce a sequence of digits from the canonical digit set, $\{0, 1, \ldots, r-1\}$, where r is the user-specified radix value. This extractor was first proposed in [1] for the binary case, $r = 2$, and was shown to maximize Shannon's entropy in the extracted string [2]. Here, we generalize the extractor by incorporating a parameterized radix value r and we analyze the quality of the extracted digit strings with varying values of r. The utility of this investigation allows users of random extractors to choose radix values that produce high-quality random digit strings.

Users of random number generators (RNG) generally rely upon one of two different forms. A pseudorandom number generator (PRNG) is a deterministic function that produces periodic sequences of length N such that strings of size n with $n < N$ have characteristics resembling those of an ideal random string. Conversely, the so-called class of "true random number generators" (TRNG) is generally based upon a physical entropy source. Examples of TRNG sources used in the past are measurements of natural phenomena such as radioactive decay [3] or measurements of superimposed quantum states [4]. When the source used in a TRNG implementation results in measured values that have a deterministic as well as a random component, the sources are characterized as weakly random and thus the measurements are generally applied to a mathematical function devised to extract the entropy or randomness from the source measurements. Furthermore, the extractor function ideally produces values that are uniformly distributed regardless of the native distribution of the source measurements. For at least these reasons, extraction functions are very important with regard to the quality of TRNG output values.

A fundamental way in which PRNG and TRNG differ is in their use of either a recurrence relation or an extractor function. A PRNG produces the next pseudorandom value by applying a recursive relation to a previously generated pseudorandom value, or alternatively, a seed value at initiation time. It is clear that PRNG are actually deterministic processes since they depend upon an initial seed value and subsequent values that are deterministically computed. In the case of TRNG, values are calculated by applying an operation to a natural phenomenon such as a random signal or other time series process to produce random numbers that are aperiodic.

The majority of past work in TRNG extractors focuses upon improvements in the production of binary string outputs. A survey of randomness extractors in both the classical and quantum realm was undertaken by Berta et al [5]. A recently proposed TRNG based on the use of two independent non-stationary stochastic process sources was shown to produce increased quality random numbers [6]. In terms of non-binary TRNGs, past results are limited, however a notable contribution is a recently proposed TRNG that generates values based on the meta-stability of a ternary valued latch [7]. This latter contribution extended prior work with TRNG based on the metastability of electronic latches [8], [9]. Our extractor allows for the production of strings over a wide variety of symbol sets with radices that vary from binary to arbitrarily high radix values that are limited only by characteristics of the source.

II. Background

The TRNG considered here is composed of an entropy source and an extractor function. We use a weakly random source in the form of timing observations between received packets in a network interface card (NIC). For the case $r = 2$, a binary string is produced by the TRNG. We first review the binary extractor function since it serves as a baseline comparison case for the higher-radix extractor. We next consider the entropy present in a non-binary string and consider how entropy varies for a given set of source observations extracted into different symbol sets of varying radix values.

A. Binary Extractor Function

It is assumed that a weakly random source produces a discrete time series $X_t = \{x_1, x_2, x_3, ..., x_n\}$ from which entropy is to be extracted. We note that a weakly random source that produces a continuous-valued X_t may also be used with an appropriate sampling function to provide discretized values. A measure of center function, $M(X_t)$, is used to find an overall centered metric value characterizing the discrete set X_t. A binary string is thus extracted through a series of comparisons of each $x_i \in X_t$ with the $M(X_t)$ value. This result of this comparison serves as the binary extraction function and is given in Equation 1.

$$f(x_i) = \begin{cases} 0, \text{ if } x_i > M(X_t) \\ 1, \text{ if } x_i < M(X_t) \\ \emptyset, \text{ if } x_i \equiv M(X_t) \end{cases} \quad (1)$$

In general, the value $M(X_t)$ can be any measure of center for a finite times series. However, the only measure of center that maximizes Shannon's entropy for a binary case is the median [1]. The median is a statistic that can be explicitly defined and is calculated differently depending on the length n of the observed time series. If the length of the series is odd, there is a value $x_m \in X_t$ corresponding to the median. However if the length of the series is even, then the median falls between two values, $(x_{1m}, x_{2m}) \in X_t$. Due to these two cases, we define $M_o(X_t)$ to be the median if the length of the series is odd, and $M_e(X_t)$ to be the median if the length of the series is even. Equations 2 and 3 contain the definitions for the median function used in the binary extractor for the n odd and n even cases respectively.

$$M_o(X_t) = \{x \in X_t \mid |\{y \in X_t \mid y < x\}| \equiv |\{z \in X_t \mid z > x\}|\} \quad (2)$$

In the even formula given below, it the case that $(x_{1m}, x_{2m}) \in X_t$. When the values of X_t are sorted into ascending order, then x_{1m} is the $\lfloor \frac{n}{2} \rfloor^{th}$ value and x_{2m} is likewise the $\lceil \frac{n}{2} \rceil^{th}$ value.

$$M_e(X_t) = \{\frac{x_{1m} + x_{2m}}{2} \mid |\{y \in X_t \mid y < x_{1m}\}| \equiv |\{z \in X_t \mid z > x_{2m}\}|\} \quad (3)$$

A single expression for $M(X_t)$ can then be formulated as given in Equation 4.

$$M(X_t) = \begin{cases} M_o(X_t) \text{ , } |X_t| (\text{mod} 2) \equiv 1 \\ M_e(X_t) \text{ , } |X_t| (\text{mod} 2) \equiv 0 \end{cases} \quad (4)$$

Using the median in Equation 4, the extractor function for a binary string as given in Equation 1 is fully specified for both the cases of $|X_t|$ being either even or odd [1]. In terms of generating strings for $r > 2$, the naive approach of up-converting the binary string to a higher radix value will cause a loss of entropy and is thus not a satisfactory solution. The extension of the extraction function to allow the production of a symbol string based upon an arbitrary base while minimizing entropy loss is a main contribution of this paper.

B. Entropy Calculation (for Higher Radix Strings)

In the Mathematical Theory of Communication by Claude E. Shannon [2], the entropy of a string $S_\Lambda = \{s_1, s_2, s_3, ..., s_n\}$ of size n based upon the symbol set $\Lambda = (\lambda_1, \lambda_2, ..., \lambda_r)$ of size r is given by formulas 5 and 6.

$$p_j \equiv P(s \in S_\Lambda = \lambda_j) = \frac{|\{s \in S_\Lambda \mid s = \lambda_j\}|}{|S_\Lambda|} \quad (5)$$

$$H(S_\Lambda) = -\sum_{j=1}^{r} p_j \cdot \log_r(p_j) \quad (6)$$

In the context of calculating the entropy for strings in different bases we imagine that the symbol set Λ consists of the required digits to construct a base-r number system. For example, consider the string S_Λ as a decimal (*i.e.*, base 10) number $S_\Lambda = (152536491023975839275912)_{10}$. S_Λ is comprised of 24 decimal digits and $\frac{1}{24} \approx 0.042$. The symbol set Λ consists of the ten decimal digits, $\Lambda = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The vector of probability values to be used in Equation 6 for the entropy calculation of the string represented by S_Λ is $\vec{p}$.

$$\begin{aligned} \vec{p} &= (p_1, p_2, ..., p_{10}) \\ &= (0.042, 0.125, 0.167, 0.125, 0.042, 0.167, \\ &\quad 0.042, 0.083, 0.042, 0.167) \end{aligned}$$

Using $\vec{p}$ and Equation 6, the entropy for the string of decimal digits represented by S_Λ can be computed as $H(S_\Lambda) = 0.93481$.

A qualitative interpretation of the entropy value is in regard to the information each digit of the original string represents or carries. In other words we can say that each digit of our original symbol string S_Λ carries approximately 93 percent of the information which it could ideally carry were it uniformly distributed.

C. Entropy Change in Strings of Differing Symbol Sets

When considering changes in entropy for a given source observation X_t, there are a couple of possibilities. One potential method is to extract X_t into an symbol set Λ or r symbols and then to consider it as a string that is represented in a larger symbol set Λ^* consisting of $r + \delta^*$ symbols. We show that this naive and straightforward approach is inferior to the alternative of extracting X_t directly into a symbol set from Λ^*. It is desirable to extract a source directly into a higher radix symbol set Λ^* rather than extracting a source into a lower radix symbol set Λ and then interpreting the extracted symbols as Λ^* strings since the resulting string, when extracted directly into the Λ^* symbol set actually has a higher entropy as compared to the naive approach. Theorem 1 quantifies the difference in entropy due to interpreting a string extracted into the Λ symbol set a Λ^* string.

Theorem 1: The change in the entropy for a given string of symbols expressed in a set Λ versus when extracted into a symbol set Λ^*, depends only upon the original symbol set size $|\Lambda| = r$, and the difference $\delta^* = |\Lambda^*| - |\Lambda|$ for $\delta^* > 0$ as given by Equation 7.

$$\Delta H = \log_{r+\delta^*}\left(\frac{r}{r+\delta^*}\right) \cdot H(S_\Lambda) \tag{7}$$

Proof 1: Consider that we have a symbol string S extracted from a source sequence X_t using a first symbol set Λ and a second string S_{Λ^*} that is also extracted from the same X_t but utilizes a second symbol set denoted Λ^*. The two symbol sets Λ and Λ^* are both countable with $|\Lambda^*| > |\Lambda|$ and $\delta^* = (|\Lambda^*| - |\Lambda|) > 0$. We modify formula 6 to account for expressing the string in the Λ^* symbol set.

$$H(S_{\Lambda^*}) = -\sum_{j=1}^{r+\delta^*} p_j \cdot \log_{r+\delta^*}(p_j) \tag{8}$$

The modifications while slight are important. We know that $\Lambda \subset \Lambda^* \implies \delta^* > 0$ because there are no observations of symbols in the set $\Lambda^* \setminus \Lambda$ in the string, $S_{\Lambda^*} = S_\Lambda$. We arrive at a situation where we have a probability of zero for those symbols in $\Lambda^* \setminus \Lambda$ since $\lim_{x\to 0} x \cdot log(x) = 0$. Hence, using this result we can show that equation 8 can be reduced to equation 9, with no difference other than the upper limit of the summation.

$$H(S_{\Lambda^*}) = -\sum_{j=1}^{r} p_j \cdot \log_{r+\delta^*}(p_j) \tag{9}$$

We now derive a closed form for the difference in entropy for the same string represented in different symbol sets by subtracting Equation 9 from 6.

$$\begin{aligned}
\Delta H &= -\sum_{j=1}^{r} p_j \cdot \log_r(p_j) - (-\sum_{j=1}^{r} p_j \cdot \log_{r+\delta^*}(p_j)) \\
&= \sum_{j=1}^{r} p_j \cdot \log_{r+\delta^*}(p_j) - p_j \cdot \log_r(p_j) \\
&= \sum_{j=1}^{r} p_j \cdot \Big(\frac{\ln(p_j)}{\ln(r+\delta^*)} - \frac{\ln(p_j)}{\ln(r)}\Big) \\
&= \sum_{j=1}^{r} p_j \cdot \ln(p_j)\Big(\frac{\ln(r) - \ln(r+\delta^*)}{\ln(r)\ln(r+\delta^*)}\Big) \\
&= \sum_{j=1}^{r} p_j \cdot \Big(\frac{\log_r(p_j)}{\log_r(e)}\Big)\Big(\frac{\ln(r) - \ln(r+\delta^*)}{\ln(r)\ln(r+\delta^*)}\Big) \\
&= \Big(\frac{1}{\log_r(e)}\Big)\Big(\frac{\ln(r) - \ln(r+\delta^*)}{\ln(r)\ln(r+\delta^*)}\Big) \cdot \sum_{j=1}^{r} p_j \cdot \log_r(p_j) \\
&= \Big(\frac{1}{\log_r(e)}\Big)\Big(\frac{1}{\ln(r+\delta^*)} - \frac{1}{\ln(r)}\Big) \cdot H(S_\Lambda) \\
&= \Big(\frac{1}{\log_r(e)}\Big)\Big(\frac{\log_r(e)}{\log_r(r+\delta^*)} - \frac{\log_r(e)}{\log_r(r)}\Big) \cdot H(S_\Lambda) \\
&= \Big(\frac{1}{\log_r(r+\delta^*)} - \frac{1}{\log_r(r)}\Big) \cdot H(S_\Lambda) \\
&= \Big(\frac{1}{\log_r(r+\delta^*)} - 1\Big) \cdot H(S_\Lambda) \\
&= \Big(\frac{\log_r(r)}{\log_r(r+\delta^*)} - 1\Big) \cdot H(S_\Lambda) \\
&= (\log_{r+\delta^*}(r) - 1) \cdot H(S_\Lambda) \\
&= \log_{r+\delta^*}\left(\frac{r}{r+\delta^*}\right) \cdot H(S_\Lambda)
\end{aligned}$$

Thus, the change in the entropy for the same string in terms of information content but when expressed in different symbol sets is only dependent on the original entropy $H(S_\Lambda)$, the original symbol set size r, and the difference δ^*. □

The result of this proof validates the intuitive result that extraction directly into a larger symbol set allows for an increased amount of entropy to be present. A larger symbol set is more expressive and allows for a greater amount of information to be present for a given set of symbols than a string of the same number of symbols but with a smaller symbol set. This is because less redundancy in terms of repeated symbols is present. This result underscores that fact that many binary extractors are in effect wasting or neglecting to capture some of the inherent entropy present in the source. Extracting an entropy-rich X_t sequence into a binary string for example, only allows a single bit of information to be extracted per observation when, in many cases, there is much more than a single bit of information per observation in a $x_i \in X_t$ source measurement.

D. Network Packet Delay as Weakly Random Source

The entropy source is an important component in any TRNG implementation. The source must be shown to depend on a quantity that can be represented by a random variable (RV). Although the parameterized extractor function described in this paper may be applied to any weakly random source, we chose a simple and readily available source for the purposes of generating our empirical results. We chose to use a time series composed of inter-packet delay values as observed in a network interface card (NIC) residing in a general purpose computer. The delays between subsequent packets received by a NIC depend upon a variety of environmental factors including the actions of the user, the current network usage, and physical effects and delays depending on the route that information is transmitted along. Representing the cumulative effects of these and other environmental characteristics as a RV has been used for a variety of network types including adhoc wireless, social, and biological such as neural or cellular types [10]–[13].

A common assumption is that network packet delay sequences are modeled as a time series with a RV that is exponentially distributed. However, there are studies that indicate other distributions may be more appropriate such as the truncated normal distribution [14]. For our purposes, the exact distribution is not important and it could even be in the form of a nonparametric distribution. The important aspect is that the packet delays contain some amount of inherent randomness or entropy. From this point of view, we assume that the observed packet delay values are modeled as a RV that has an arbitrary probability mass function (PMF) denoted as $P_{pack}(x_i)$. The objective of the extractor function is then to extract randomness from the observed packet delay sequence such that the resulting PMF of the extracted values is as close as possible to a theoretical uniform distribution.

In this particular case, we use the flow of information across a network as our weakly random source. As the information flows across a network point A to B, it takes many different routes depending on traffic congestion as well factors such as router and server maintenance *etc.* This causes random variations in the arrival times of packets. Measuring these precise arrival times, and reducing the data by looking only at arrival time differences or the first order features, is done automatically by NICs and can be reported using simple tools like tcpdump and Wireshark. Unfortunately the timing elements on the network interface card do not have infinite precision and are hence limited in their resolution of measured time values. The resolution restriction affects the quality of the random values produced since a higher resolution allows the radix value to increase. There is a practical upper limit to the radix value due to the resolution of the inter-packet delay observations as will be more fully discussed in a later section of this paper.

To obtain the inter-packet delay values from the NIC, we use the Wireshark packet capture utility to obtain the packet time stamps, t_i, from the NIC. After receiving the t_i values, the corresponding inter-packet delays are computed as $x_i = t_{i+1} - t_i$. Wireshark reports a packet arrival time with a time stamp value that is the number of microseconds that have elapsed since the last second of network epoch time. Network epoch time is the total number of seconds that have elapsed since January 1, 1970. Thus, the timing resolution of the packet arrival times reported by Wireshark is one microsecond, 1×10^{-6} seconds.

III. Extractor Function with Parameterized Radix Value

The theory behind extracting random values from a weakly random sequence into a non-binary symbol sequence can be considered from the point of view of the amount of information each non-binary symbol represents. In the case of the binary extractor function, a single test was performed. Specifically a test that considers whether an observed value is greater than or less than a measure of center. Therefore, in the case of the generalized extractor, we wish to evaluate $r-1$ tests in order to determine a single symbol. Thus, the amount of information each symbol represents is greater than that of the binary case since more information is derived from the outcomes of a greater number of tests. Figure 1 depicts the generalized flow of the extraction process as proposed in this research.

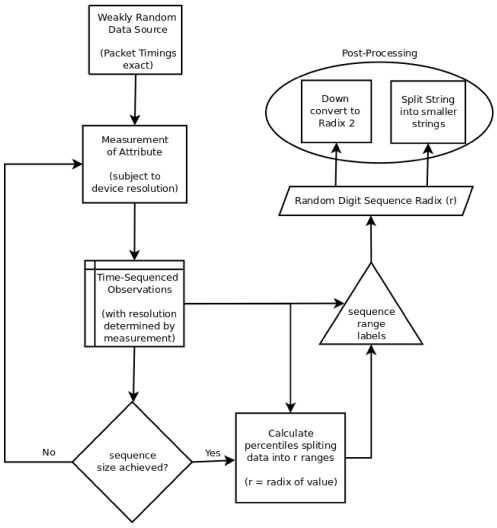

Fig. 1. A Multi-valued TRNG Methodology

Presumably the desired output string size and the radix of the output string is a priori knowledge provided by the user. When the desired sequence size n has been reached, the appropriate percentiles for evenly dividing the data range into r equivalent ranges has been calculated and the data is thus sequenced based upon the order in which it was observed. For instance, if the first data point falls within the i^{th} range then the i^{th} symbol of the symbol set $s_i \in \Lambda$ is recorded at the current position. At the next position, the subsequent observation's position is determined and the corresponding information is extracted into the next digit, and so forth, until all observations have been sequenced and the random string is produced.

Based upon the considerations expressed above, the multiple-valued extractor function is formulated as a direct generalization of the binary extractor described in Section II-A. To extract random values from a weakly random sequence of network packet spacings, we first divide the sample space into r partitions with each partition defined as $(m_{i-1}(X_t), m_i(X_t))$. The partition set has cardinality $r+1$ and is expressed as $\{m_0, m_1, m_2, \ldots, m_i, \ldots, m_{r-1}, m_r\}$. Next we process the observed packet delay values in the order in which they were observed and output a symbol s_i where the integer index i has a range of $[0, r-1]$ and depends upon which partition each observed delay value falls within. In the work described here, we use the canonical symbol symbol set where $s_i = i$ although any arbitrary symbol set could be used. This process results in an extractor function that is parameterized with radix value r causing the weakly random source observations to yield symbols from the symbol set

$s_i \in \{0, 1, \ldots, r-1\}$ for each observed packet delay value x_i. The extractor function is expressed mathematically as shown in Equation 10.

$$f(x_i; r) = \begin{cases} \emptyset | x_i = m_{i-1} \\ s_i | (m_{i-1}(X_t) < x_i < m_i(X_t)) \\ \emptyset | x_i = m_i \end{cases} \quad (10)$$

IV. Implementation and Experimental Results

Given sequenced observations from a weakly random source $\{X_t\}$ of length n, we provide an algorithm that can be used to extract a random value into a symbol sequence of maximum length n, with radix r. This algorithm implements the extractor function given in Equation 10. The function "Quantiles(m,X_t)" returns the quantiles of X_t partitioning the data set X_t into m equally sized regions.

Result: Extract random values from random source
Input : X_t, Radix r
Output: R, symbol vector

$R \leftarrow \{\emptyset\}$
$Q \leftarrow$ Quantiles$(r-1, X_t)$
foreach $x \in \{X_t\}$ **do**
  foreach $q \in Q$ **do**
    if $x \geq q$ **then**
      break
    end
    if $x < q$ **then**
      $R \leftarrow \{R, \text{Index}(q, Q)\}$
      break
    end
  end
end

Algorithm 1: Arbitrary Radix Extraction

The algorithm is implemented using the R statistical programming language and the behavior of produced random values given different extraction radices is examined. To visually illustrate the operation of the extraction algorithm, we used a standard Mersenne twister-based PRNG that simulates a discretized standard normal distributed RV to obtain an example X_t series and ran the algorithm with radix values of $r = 2, 3, 4$. The X_t sequence consists of the same eight points for each radix and was obtained using the "rnorm" function in R with a default mean of zero, a unity variance, and a seed value of `0xFEED`. In the plots shown in Figure 2, the computed quantile boundaries are indicated by colored horizontal lines. Each point in X_t is indicated by the small circle in the plots and the corresponding extracted sysmbol is shown in colored font near the top of the plot. It is noted that the inter-quantile spacings are not the same since they depend upon the range of values in X_t. These non-uniform quantile intervals are necessary to transform the native X_t PMF, $P_{norm}(x_i)$, into the desired uniform PMF and is the essence of the extractor methodology.

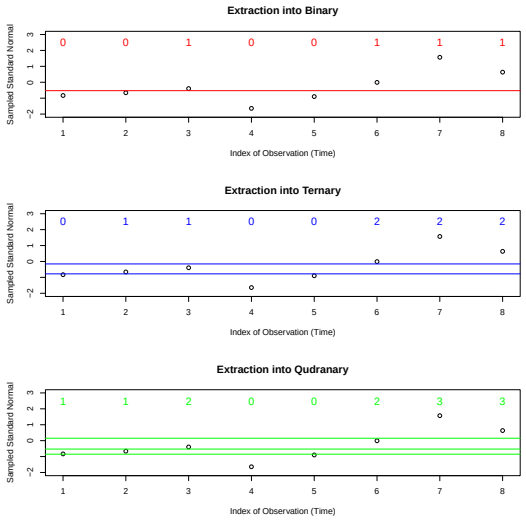

Fig. 2. Extracting Random Values into Higher Radix Strings

V. Experimental Results

To evaluate the effectiveness of the parameterized extractor function, we constructed a TRNG using the weakly random source described previously and extracted symbol strings from the source data. We used the same set of source data and extracted symbol strings with varying radix values. Since the objective of a TRNG is to produce data that is uniformly distributed, we compared the sample mean and variance of the extracted strings to that of a theoretical uniform distribution over the unit interval. The theoretical mean of a perfectly uniform distribution over the unit interval is $\mu = \frac{1}{2}$ and the corresponding variance is $\frac{1}{12} \approx 0.08333$.

Our experimental data consists of an ensemble of packet delay sequences X_t with each sequence consisting of one hundred packet spacings. Our ensemble consists of a total of 448 X_t sequences corresponding to a total number of 44,800 packet delay values. Therefore, we obtained sample mean and variance values for each of the 448 X_t sequences for each radix value considered. Each X_t sequence resulted in an extracted string that consists of a maximum of 100 symbols, although the exact number of extracted symbols is typically slightly less than 100 since packet delay values that were equal to quantile boundary values were discarded. The reason that packet delays which were equal to quantile boundary values are discarded is to avoid the introduction of a bias into the extracted symbols. If we were to assign such delay values to the symbol corresponding to the quantile just below boundary value, a negative bias would be present in the mean. Likewise, assigning the delay value to a symbol corresponding to the quantile range greater than the boundary value would introduce a positive bias to the mean.

Figure 3 contains a plot of the average of each X_t sample mean versus the radix value used in the extractor. allowable value of the radix due to the resolution of the packet delay

measurement. When the radix value of the extracted sequence exceeds the reciprocal of the measurement resolution, no benefit in continuing to increase the radix value will result. Furthermore, as the number of quantiles increases due to an increasing radix value, a large number of packet delay samples will exactly equal the quantile boundary causing those points to be discarded. At the extreme case where the radix value equals or exceeds the reciprocal of the measurement resolution, all values will equal a quantile boundary causing every point to be discarded from the extracted sequence. Based on these observations, a tradeoff is apparent in that the randomness quality of the extracted sequence tends to increase proportionally with the radix value, however, the number of extracted symbols begins to decrease as the radix value approaches the reciprocal of the measurement resolution.

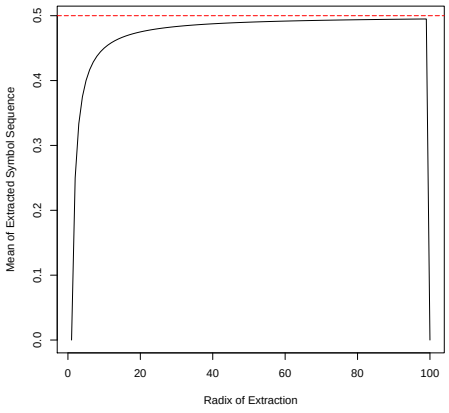

Fig. 3. Average Sample Mean of 448 Extracted X_t Symbol Sequences

Figure 4 contains a plot of the average variance of each extracted X_t sequence as the radix increases. The ideal variance, $\frac{1}{12}$, is indicated by the horizontal red line on the plot. The localized changes of average variance are accounted for by considering the changes in the number of discarded points from one radix value to another. When the quantile boundary values are perfect multiples of the measurement resolution, there is a higher likelihood that sample values will equal a quantile boundary and thus cause more values to be discarded.

VI. Conclusion

An entropy extraction function has been introduced that is based upon a parameterized radix value. The radix parameter allows the extractor to produce strings using different symbol sets of varying sizes for a given weakly random source. We have also provided theoretical results regarding the entropy present in strings composed of higher-radix, non-binary symbol sets and derived expressions that indicate how the entropy can vary. Our results provide guidelines that allow a TRNG designer or user to choose an appropriate radix value based upon the inherent characteristics of the weakly random source such as measurement resolution. We have introduced

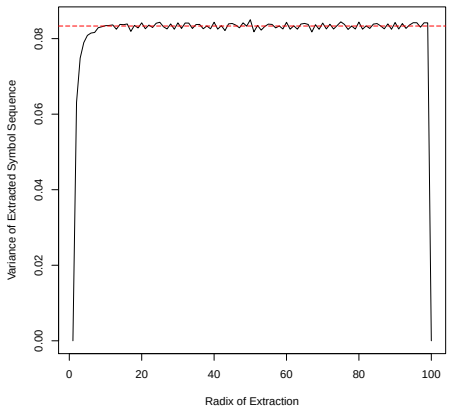

Fig. 4. Average Sample Variance of 448 Extracted X_t Symbol Sequences

the concept of TRNG extraction efficiency and conjectured that many binary extractors are losing source entropy.

References

[1] M. Thornton, "Randomness properties of cryptographic hash functions," Master's thesis, Dept. of Computer Science and Engineering, Southern Methodist University, 2017.

[2] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 4, pp. 623–656, Oct 1948.

[3] A. Figotin, P. Vitebsky, G. Stetsenko, S. Molchanov, A. Gordon, J. Quinn, and N. Stavrakas, "Random number generator based on the spontaneous alpha-decay," US Patent 6,745,217, June 1, 2004.

[4] W. Dultz, G. Dultz, E. Hildebrandt, and H. Schmitzer, "Method for generating a random number on a quantum-mechanical basis and random number generator," US Patent 6,609,139, August 19, 2003.

[5] M. Berta, O. Fawzi, V. Scholz, and O. Szehr, "Variations on classical and quantum extractors," in *2014 IEEE International Symposium on Information Theory*, June 2014, pp. 1474–1478.

[6] G. Martini and F. G. Bruno, "True random numbers generation from stationary stochastic processes," in *2017 International Conference on Noise and Fluctuations (ICNF)*, June 2017, pp. 1–4.

[7] S. Tao and E. Dubrova, "Tvl-trng: Sub-microwatt true random number generator exploiting metastability in ternary valued latches," in *2017 IEEE 47th International Symposium on Multiple-Valued Logic (ISMVL)*, May 2017, pp. 130–135.

[8] V. B. Suresh and W. P. Burleson, "Entropy and energy bounds for metastability based trng with lightweight post-processing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 7, pp. 1785–1793, July 2015.

[9] S. Walker and S. Foo, "Evaluating metastability in electronic circuits for random number generation," in *Proceedings IEEE Computer Society Workshop on VLSI 2001. Emerging Technologies for VLSI Systems*, May 2001, pp. 99–101.

[10] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 2, no. 3, pp. 257–269, July 2003.

[11] G. Robins, P. Pattison, Y. Kalish, and D. Lusher, "An introduction to exponential random graph (p*) models for social networks," *Social networks*, vol. 29, no. 2, pp. 173–191, 2007.

[12] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless communications and mobile computing*, vol. 2, no. 5, pp. 483–502, 2002.

[13] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[14] A. Sukhov and N. Kuznetsova, "What type of distribution for packet delay in a global network should be used in the control theory?" *arXiv:0907.4468v1 [cs.NI]*, p. 4, July 2009.