Title: How Random is Your RNG?

Primary Author Name:	Meltem Sonmez Turan
Primary Author Affiliation:	NIST
Primary Author Email:	meltem.turan@nist.gov
Additional Author Name:	John Kelsey
Additional Author Affiliation:	NIST
Additional Author Email:	john.kelsey@nist.gov
Additional Author Name:	Kerry McKay
Additional Author Affiliation:	NIST
Additional Author Email:	kerry.mckay@nist.gov

Keywords/Tags: entropy, random number generation, RNG, prediction

Abstract:
Cryptographic primitives need random numbers to protect your data. Random numbers are used for generating secret keys, nonces, random paddings, initialization vectors (IVs), salts, etc. Deterministic pseudorandom number generators are useful, but they still need truly random seeds generated by entropy sources in order to produce random numbers. Researchers have shown examples of deployed systems that did not have enough randomness in their entropy sources, and as a result, crypto keys were compromised. So how do you know how much entropy is in your entropy source?

Estimating entropy is a difficult (if not impossible) problem, and we've been working to create usable guidance that will give conservative estimates on the amount of entropy in an entropy source. We want to share some of the challenges and proposed methods. We will also talk about some new directions that we're investigating, and present the results of our estimation methods on simulated entropy sources.

Introduction

Random numbers are necessary for cryptography. Every time someone generates a key, initialization vector, nonce, random signature parameter, etc., they use a random number generator. The strength of those random numbers is important--there have been real-world attacks allowed by failures to generate cryptographically strong random numbers [1].

NIST Special Publication (SP) 800-90 (a series consisting of three documents) is all about generating random numbers for cryptography. In SP 800-90, this is a two-stage process: first, an *entropy source* provides an impossible-to-guess *seed*. Then, a deterministic cryptographic algorithm (called a DRBG--deterministic random bit generator--in SP 800-90) expands the seed into a long sequence of values that may be safely used for keys, IVs, nonces, etc.

The entropy source must be based on some nondeterministic process--for example, many practical entropy sources are based on the timing variation from an unstable oscillator. To be useful, the entropy source's unpredictability must also be quantified--we need to know how many bits need to be drawn from the entropy source to produce a good seed. The unpredictability of the outputs of an entropy source is measured in terms of *entropy*.

There are a number of different measures of entropy; for seeding a DRBG, the relevant measure is called *min*-entropy, which corresponds to the difficulty of guessing the most-likely output of the entropy source. If $\boldsymbol{p}$ is the probability of the most likely value for some variable, then its min-entropy is $H = -\lg(p)$.

The 90B Non-iid Estimators

Draft SP 800-90B (one of the SP 800-90 documents, and called 90B from now on) discusses procedures for evaluating how much entropy per sample can be obtained from an entropy source [2], and includes five entropy estimators[1] for non-iid[2] data. Each estimator takes a sequence of minimally processed samples from the underlying unpredictable process in the entropy source, and uses them to compute an entropy estimate. The minimum of these five estimates is used as the entropy assessment of a given source.

Issues with the 90B Estimates

There are some issues with the 90B estimates, which justify the search for better ways to estimate entropy.

a. Four of the five 90B estimates have to assume that the source is iid in order to compute their entropy estimate. Since they are only expected to be used on non-iid sources, this is problematic.
b. The non-iid estimates tend toward underestimates, and some of them will give huge underestimates on some kinds of sources. This interacts badly with the strategy of estimating the entropy of a source by taking the minimum of all the estimates.

Predictors: An Alternative Approach to Entropy Estimation

A *predictor* is an algorithm that aims to predict the next sample value, based on previous observations. Entropy is a measure of unpredictability, so we can use the accuracy of a predictor to estimate the entropy of a source. The more accurate the predictor's predictions, the lower the entropy estimate. Estimates are calculated in two ways:

a. Based on the predictor's average performance, using the number of correct predictions divided by the number of attempted predictions.
b. Based on the predictor's best burst of performance, using the longest run of correct predictions.

---

[1] These five estimators were contributed to 90B by the National Security Agency (NSA), and are discussed in [3] by Hagerty and Draper.

[2] Sources in 90B are split into iid and non-iid sources. An iid (*independent and identically distributed*) source is a source whose samples are all drawn from the same distribution, and are all independent of surrounding samples. A non-iid source is any source that isn't iid. In general, it's easy to estimate the entropy for iid sources, and harder for non-iid sources.

The final entropy estimate is the minimum of these two values.

Predictors have some major advantages over the 90B estimates. A predictor that is a good fit for the behavior of a source will give a relatively accurate estimate; a predictor that is a poor fit will give an overestimate. This interacts well with the procedure of taking the minimum of many entropy estimates to get a final estimate. Adding many predictors, each tuned for a different kind of source, will not degrade the quality of the final entropy estimates we get. More fundamentally, the 90B estimates are based on constructing a probability model and estimating the parameters of the distribution within the model, and their results are only very loosely related to any practical security question. By contrast, predictors attempt to answer the only really relevant question about an entropy source: how easy is it to predict the source's outputs?

In this research, we introduce three predictors:

a. The *Most Common in Window* (MCW) predictor maintains a window of the last *w* samples and predicts the next sample value to be the most common sample value within the window. The window size was selected to be 64 for our experiments.
b. The *Multi Markov Model with Counting* (MultiMMC) predictor keeps track of multiple Markov models simultaneously and uses the highest performer so far to make predictions.
c. The *Lag* predictor keeps track of multiple lags simultaneously and uses the highest performing lag for the prediction. Lags up to 128 were used for the experiments.

These three predictors cover three relatively simple ways that a source might exhibit some predictability. All three predictors have some additional parameters, which were set to apparently sensible values for our experiments.

Testing the Estimators with Simulated Sources

Evaluating the accuracy of entropy estimates is challenging, because real-world entropy sources do not actually come with known-correct entropy/sample numbers. The best solution we could find for this problem was to produce simulations of entropy sources. These simulated sources follow known probability distributions, and so have known entropy/sample. Samples are integers drawn from sources belonging to one of five distribution families:

a. Uniform -- all values have equal probability.
b. Near-uniform -- all values but one have equal probability; the remaining value has a higher probability than the rest.
c. Normal -- samples are drawn from a normal distribution and rounded to integer values.
d. Time-varying normal -- samples are drawn from a normal distribution and rounded to integer values, but the mean of the distribution moves along a sine curve to simulate a time-varying signal.
e. Markov models -- samples have a fixed probability distribution based on the previous *K* samples that have been seen.

For each of these simulated sources, datasets of 1 000 000 samples were generated and evaluated by both the 90B estimators, and by our predictors. We were able to track the performance of the different estimates vs. the correct entropy values.

Results

The results of our experiments were as follows:

a. The predictors were substantially more accurate than the 90B estimates. The MultiMMC predictor was especially accurate.
b. The 90B estimates gave huge underestimates for many sources.
c. The 90B estimates generally tend to underestimate our simulated sources, but in some cases (particularly Markov model sources) gave some significant overestimates.

Conclusions and Future Work

The non-iid entropy estimates from the current draft of 90B represent a serious attempt to come up with practical general-purpose entropy estimates for entropy sources. Both analytically and experimentally, however, they have some problems that call their usefulness into question. We investigated a completely different mechanism for entropy estimation, based on *predictors*, and experimentally showed that three simple predictors can outperform the 90B estimates on a variety of fairly simple simulated entropy sources.

Future research should identify more predictors that could be added, investigate the best parameter choices for the predictors presented here, and evaluate both the 90B estimates and the predictors against additional simulated sources.

References

[1] N. Heninger, Z. Durumeric, E. Wustrow and J. A. Halderman, "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices," in *Proceedings of the 21st USENIX Security Symposium*, 2012.

[2] E. Barker and J. Kelsey, *NIST Draft SP 800-90B Recommendation for the Entropy Sources Used for Random Bit Generation*, Available online http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf, 2012.

[3] P. Hagerty and T. Draper, "Entropy Bounds and Statistical Tests," in *Random Bit Generation Workshop*, 2012.