Hardware Random Number Generators

Bjørn Mathias Helseth

Royal Holloway, University of London

Table of Contents

Hardware Random Number Generators

Hardware Random Number Generators are an essential part of any secure cryptographic system. This is due to the generators ability to produce random numbers which are truly non-predictable. Using a system that is pseudo-random, usually software-driven, the generator can be vulnerable to attacks as the numbers are driven by an external factor that is predictable. A Hardware Random Number Generator, also known as a True Random Number Generator (TRNG) usually benefits from an unpredictable physical event happening.

There are a number of different use cases for RNGs and in cryptographical settings the need for these to be cryptographically secure can be crucial. Both for the end-user and the manufacturer. One of the use cases is in authentication protocols. Here the use of unbiased, unpredictable and entropically rich random numbers are essential to provide a safe platform with little-to-no risk that an adversary might be able to decrypt random keys produced and fed into the system for authentication purposes.

Some of these TRNGs works by utilizing a preferably entropically rich source to for example produce some type of breakpoint in a running circuit (either software-based or hardware-based). This can be as easy as having the breakpoint return some binary value. For example, if you receive a breakpoint from the source it is a 1, and if there are no inputs from the source that is a 0. With this we are left with a binary key which we can put into a whitening function to remove bias from the entropy stream. An example of a whitening function can be to perform an XOR operation on the stream. There also exists a number of other whitening functions and TRNGs.

Looking further into how these RNG's work, it could be possible to establish a sense of how we may analyze their output in a lightweight statistical manner to ensure that we do in fact have a truly random output with no obvious bias or a low entropically filled pool. Can we see a clear difference in what a TRNG and a PRNG are able to output?

In a PRNG we can usually find that the entropy pool can fill up or stop being refreshed with new input data. This is often the case that makes PRNGs less cryptographically secure. If we make an assumption that we have the entropy pool in one PRNG fill up with user input (clicks, mouse movement and keystrokes). This can at any point stop being fed new input, and therefore the generator will have to take its inputs from the same entropy source. This can lead to the generator outputting sequences or simply repeating the same patterns.

In a TRNG we usually find different sources of entropy used. These are physical and natural events happening at an unpredictable rate creating 'noise'. The most commonly used entropic sources in use for TRNGs as I have found on the market are among others, decay of radioactive materials, thermal noise and atmospheric noise. These allow the generator to be fed with constant noise from a source found to be unpredictable.

Currently, on the market there are a number of different TRNGs that are being sold, both privately and by corporations manufacturing them. The majority of these TRNGs are marketed as being able to pass all the top industry standard statistical tests such as Dieharder and FIPS 140-2 just to mention some of them. This seems to be their main selling point in addition to their output speed. Now, can we trust these statements without actually testing this ourselves? How can we know that there is actually some cryptographically secure module

inside the enclosure of this physical device that will be able to provide us with such high-level of entropy and randomness?

These industry standard tests are usually performed by the manufacturer, which provides the results. How can we however know that they haven't been changed after the tests, can we trust the results ourselves and could someone have tampered with the device before it got to the end user? These are vulnerabilities that can be found in most TRNGs purchased online. Could lightweight statistical tests of randomness be more available to the end-user so this can be tested more efficiently and easily?

The main reason why we can't trust pure PRNGs for our cryptographic applications is due to the entropy pool being retrieved from a source known to be predictable. For example, if we look into the /dev/urandom feature on most Linux distributions. It is provided through user input and other factors which are happening to fill up the entropy pool. However, what makes /dev/urandom cryptographically less secure than for example /dev/random, is because it does not block the generator from proceeding to output numbers even though its entropy pool is filled up. Which can make up sequences which are repetitive and therefore not random.

The clear benefit of these hardware random number generators is their ability to serve as a secure gateway to cryptographically secure applications. In my final project report I hope to use the findings through my research on these HRNGs to develop a sense of what can be done to ensure that we are dealing with a RNG that has not been tampered with and has a highly entropical pool which constantly is able to be refreshed.

Performing statistical tests of randomness on RNGs can tell us more about the characteristics of the generator in question and can help determine whether or not the RNG is suitable for use in applications as a secure way of deliver randomness. Helping the end-user performing these tests can lead to any adversary to not be able to penetrate an application that could have otherwise been easily intrudable through deterministic randomness.

The statistical tests of randomness may only act as a marker to see if the generator outputs entropically rich data and if the output does not reflect sequences. What can be expected to be seen in HRNGs, are data that do not repeat and do not follow any obvious pattern. These are usually standardized tests such as Dieharder and FIPS 140-2. The FIPS 140-2 tests are used as a US government standard to ensure cryptographic security within their applications.

Generation of random numbers through an external physical event happening is absolutely the most secure way to generate these random numbers, as an adversary will not be able to predict the numbers generated. However, the adversary might be able to bias the entropy source to perform with a distribution that is predicted to for example output numbers with less entropy by moving the entropy source itself. This however is not likely to happen in a stationary private TRNG.

Bibliography

Davies, R., 2000. Hardware random number generators.

Dichtl, M., 2003. How to Predict the Output of a Hardware Random Number Generator.

Hurley-Smith, D., Patsakis, C. & Hernandez-Castro, J., 2020. On the unbearable lightness of FIPS 140-2 randomness tests.

Rukhin, A. et al., 2010. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. *NIST Special Publication 800-22,* Issue Revision 1a.

Sönnerup, J., 2019. *Debricked.* [Online]
Available at: https://debricked.com/blog/2019/04/12/difficulty-of-generating-randomness/
[Accessed October 2020].

Taylor, G. & Cox, G., 2011. Behind Intel's New Random-Number Generator. *IEEE Spectrum.*