SIGCHI Conference Proceedings Format

Love Arreborn
lovar063@student.liu.se

Nadim Lakrouz
nadla777@student.liu.se

1 INTRODUCTION

In cryptography, there are many applications for randomly generated numbers. However, the process of producing these random numbers tends to be pseudo-random, e.g. utilizing the current states of various modules [@randomness]. These numbers do not generate true randomness, and in order to heighten security other methods of generating these are required. Current random number generators (*RNG*) are usually implemented in code, using certain states of the host machine as a starting point before running a predetermined algorithm [@randomness]. This pseudo-random number generation (*PRNG*) comes with the drawback that the result is always deterministic, provided that the initial state is known.

True random numbers, then, cannot be produced solely through code. These systems require some input that is neither replicable nor reproducible. One proposed solution for this is quantum random number generation (*QRNG*) [@QRNG]. By reading quantum fluctuations from any given source, for instance an optical signal, the inherent natural unpredictability of said source can be harnessed in order to produce a random number from a state that is nigh impossible to reproduce accurately.

In this project, we will be writing firmware for one such solution, which reads quantum variations from an optical signal. Further details about how this signal is produced will be introduced in section 2 and builds on the work of Clason [@Clason2023]. This optical signal will be converted to a stream of random, raw bits via an Analog to Digital Converter (*ADC*). In turn, these random bits will be processed via Toeplitz-hashing [@toeplitz] in order to process these bits into random numbers. Some processing has to be done on the microcontrollers themselves in order to ensure that the data is workable, and Toeplitz-hashing is a tried and tested method to accomplish this. These random numbers will then be output from the microcontroller to the host computer via USB. This thesis will aim to answer one key research question: How can sampled vacuum fluctuations be processed efficiently in order to output QRNG?

In producing this firmware, several key considerations have to be made in order for this system to be usable in a production environment. The vision for the end product is a simple USB-stick that can be connected to a host, and produce true random numbers from ambient quantum fluctuations. While this system could very well be implemented on physically larger hardware, thus avoiding the constraints that limited hardware introduces, Clason proposes a simpler and cheaper way to achieve QRNG [@Clason2023]. In keeping with this, utilizing microcontrollers for processing the raw bits extracted further helps to keep the costs low and the solution reasonably complex. Further details regarding the microcontrollers used in our work will be outlined in section 3. However, due to this portability constraint, our implementation needs to work quickly and efficiently on resource constrained hardware. As such, our main question is broken down into two concrete research areas:

Research area 1 (*RA1*): How can Toeplitz-hashing be implemented as effectively as possible on resource constrained hardware in order to process raw bits into a workable random number?

Toeplitz-hashing been optimized quite well, and previous research can be utilized for this. However, there are still considerations when implementing the firmware for the microcontroller in order to optimize the code. Our goal is to attempt several implementations in order to find the most optimal implementation with the least amount of CPU-cycles.

Research area 2 (*RA2*): How can we ensure that the output of random numbers is not limited by our firmware, but rather only limited by the USB transfer speed of the microcontroller or the ADC?

There will unequivocally be a bottleneck for the processing speed. For instance, the speed at which the ADC can process the optical signal into raw bits as well as the speed that the USB output can transfer processed random number to the host computer will be limiting factors. Further details on the limitations of the ADC will be outlined in section 3. The slowest of these bottlenecks will inevitably be the limiting factor for any implementation. Our research aims to ensure that our implementation of Toeplitz-hashing does not become the limiting factor, but rather processing data fast enough to match or exceed the speed of the hardware.

2 RELATED WORKS

This work is a practical continuation of the work of Clason [@Clason2023]. In this work, the author aimed to study quantum shot noise originating from photodiodes, and in so doing built a device which read from an optical source, outputting analog voltage from the data "seen" by the diodes. A prototype was constructed, in which an LED is read by a photodiode soldered millimeters apart.

The idea of an optical QRNG (*OQRNG*) is not a novel one. The basis of the theory is that intrinsically random properties of a quantum process. Stefanov et. al. proposes using the random choice of a photon between two output signals to generate a random stream of bits, however the theory behind it can be applied to other quantum processes as well [@StefanovOptical]. This particular theorem has been implemented by Wayne et. al. to create a quantum number generator [@Wayne]. While this article proves the efficacy of OQRNG, it utilizes a slightly different method.

Our work revolves around the measurement of shot noise of vacuum states rather than measuring arrival times of photons. Essentially, this is another quantum process with the same inherently random properties as described by Stefanov et. al., but instead using shot noise. As described by Niemczuk [@shotnoise], shot noise is minor fluctuations in an elecritcal current, which is inherently random. Reading this property, then, gives us an intrisically random source from which to generate a random output, which in turn can be processed into a random number.

Implementations of this theory exist, however with significant drawbacks. Shen et. al. presents an implementation using a fairly complex setup, in which a continuous-wave fiber laser is the optical source [@contender1]. They conclude that sampling the shot noise is, indeed, suitable for OQRNG. However, the implementation requires expensive and complex hardware, and the sheer size of the system prohibits it from being portable and easily reproducible in small-scale tests.

A more recent implementation of OQRNG in a smaller scale has been presented by Singh et. al. [@singh]. This particular implementation uses a bespoke circuit board where all components are present on a single board – e.g., this experimental setup contains an integrated ADC, post-processor, entropy controller and entropy generator. While this article cements the viability of OQRNG using shot noise (despite the article not being confirmed as peer reviewed), the bespoke nature of the circuit board makes this experiment difficult to reproduce. As our thesis will use commercially available ADCs and microcontrollers, the only bespoke component is the shot noise generator itself. Furthermore, the Toeplitz-hashing is not run on the microcontroller itself in these experiments – instead, the hashing of these raw bits is done on the receiving computer as this bespoke circuit board featured a relatively weak processor.

In summary, previous research has proven that OQRNG can generate true randomness, and more specifically, Shen et. al. [@contender1] and Singh et. al. [@singh] both implement OQRNG through readings of shot noise. However, there are limitations in both of these works. Either the system that generates the shot noise is large and complex [@contender1] or the system is built on bespoke hardware with limitations in processing power which prevents a fully integrated system [@singh]. Our work aims to bridge this gap by using commercially available hardware (other than the bespoke shot noise generator [@Clason2023]) and focuses on implementing Toeplitz-hashing directly on the microcontroller.