Journal of Modern Optics

ISSN: 0950-0340 (Print) 1362-3044 (Online) Journal homepage: www.tandfonline.com/journals/tmop20

Photon arrival time quantum random number generation

Michael A. Wayne, Evan R. Jeffrey, Gleb M. Akselrod & Paul G. Kwiat

To cite this article: Michael A. Wayne, Evan R. Jeffrey, Gleb M. Akselrod & Paul G. Kwiat (2009) Photon arrival time quantum random number generation, Journal of Modern Optics, 56:4, 516-522, DOI: 10.1080/09500340802553244

To link to this article: https://doi.org/10.1080/09500340802553244

Published online: 18 Mar 2009.

Journal of Modern Optics
Vol. 56, No. 4, 20 February 2009, 516–522

Taylor & Francis
Taylor & Francis Group

Photon arrival time quantum random number generation

Michael A. Wayne*, Evan R. Jeffrey, Gleb M. Akselrod and Paul G. Kwiat

Department of Physics, University of Illinois at Urbana-Champaign, 1110 W. Green Street, Urbana, IL 61801, USA

(*Received 16 April 2008; final version received 11 October 2008*)

We present an efficient random number generator based on the randomness present in photon emission and detection. The interval between successive photons from a light source with Poissonian statistics is separated into individual time bins, which are then used to create several random bits per detection event. Using a single-photon counter and FPGA-based data processing allows for a cost-efficient and convenient implementation that outputs data at rates of roughly 40 Mbit s^{-1}.

Keywords: quantum random number generation; photon counting; quantum cryptography

1. Introduction

The need for randomness arises frequently in a broad spectrum of applications, ranging from numerical simulations and statistical analysis to encryption. Methods for achieving this randomness have advanced as the applications increase, evolving from simple mathematical techniques which generate pseudo random numbers to physical sources of true random numbers. Dependent on the constraints of the application involved, it is often sufficient to use numbers that are not actually random, but random *enough*. These can be realized, e.g., by a pseudo random number generator which performs mathematical operations on data based on a seed, typically a very large number. Unfortunately, such schemes are deterministic: the same initial seed will always create the same sequence of pseudo random numbers. As computing power increases, this is no longer sufficient for many cryptographic applications, as it is possible for these algorithms to be compromised. In particular, one of the promises of the recent area of quantum cryptography is unconditional security based on laws of physics [1]. For such applications it is critical to have true sources of randomness.

A quantum random number generator (QRNG) exploits the inherent randomness present in quantum processes to create random numbers. Common implementations are based on the interaction of photons with a beam splitter [2–4], where a random bit is determined by which path a photon takes. More recently, it was realized that one can use the *time* between successive photons in a single path to generate randomness [5–7].

We start by explaining the theory underlying our design, as well as detailing its implementation. The performance of our system is then discussed, including the tests performed to determine the amount of randomness present in our data. Finally, we briefly discuss proposed improvements for future designs.

2. Theory of operation

Several QRNGs based on the quantum properties of light have been proposed and implemented. Most previous systems [2,3,8,9], such as the one depicted in Figure 1(*a*), rely on the behavior of an incoming photon at a beam splitter to generate data. Dependent on which detector receives the incoming photon, a '0' or a '1' is generated. This approach has the significant drawback that each photon can create at most only one bit of data, and in practice much less, since it is only *detected* events that contribute. Thus, the scheme is limited by the detection speed: one can only generate random bits at rates substantially below the detector saturation limit. While somewhat mitigating this problem, implementations with *multiple* detectors can suffer from bias created by differing detection efficiencies. Recently, it was shown [7] that such bias could be eliminated by using a single detector, and comparing the time intervals between three successive detection events. However, this method is limited to a maximum of one-half bit of randomness per detection, so is even more constrained by detector saturation.

Our implementation [6] also uses only a single detector to generate the data, but uses the photon arrival time itself as the quantum random variable, as

*Corresponding author. Email: mwayne@uiuc.edu

ISSN 0950–0340 print/ISSN 1362–3044 online

DOI: 10.1080/09500340802553244
http://www.informaworld.com

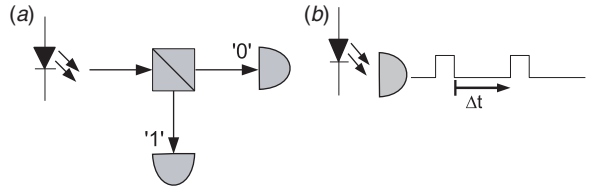

Figure 1. Examples of QRNG implementations. Photons in (*a*) are directed along one of two paths by a beam splitter, and are registered by the corresponding detector. In (*b*), time intervals between successive registered photons are translated into time-bin values. Because the average number of time bins between detections can be quite large, one can distill multiple random bits per detection event.

shown in Figure 1(*b*). As originally proposed [5,6], the time between successive photons is divided into 'time bins', which are created by a high-resolution counter operating in parallel with the detector (in principle this could also be combined with a beam splitter and two detectors to obtain an extra random bit per detection). A given detection time interval can therefore provide multiple random bits per detection event. If we had a *constant* waiting-time distribution, for which every time bin had the same probability of occurring, then, given n bins, we would generate $\log_2(n)$ random bits per detection event. However, the waiting-time distribution for a Poisson process is a decaying exponential, as shown in Figure 2 (red curve). Specifically, after a detection event, it is more likely that the next detection event will occur sooner rather than later, so if the interval is broken into equal-sized time bins, the earlier ones are more likely than the later ones. Consequently, the resulting detection events will not be completely random; more precisely, the randomness associated with each detection will be less than that if the distribution were simply uniform. In order to compensate for this lack of randomness, a data-hashing technique must be used to 'whiten' the random number string, thereby preparing a shorter but more random string, with randomness approaching one random bit per bit. (Alternatively, one could use a shaped optical pulse to approximate a constant waiting-time distribution [5,6], though here too it is likely that *some* residual hashing would be needed.)

The random entropy of an arbitrary waiting-time distribution can be calculated from the Shannon entropy, defined as

$$S = -\sum_{i=0}^{N} P_i \log_2 P_i, \tag{1}$$

where P_i is the probability of a detection event corresponding to time bin i, summed over all possible bins N. This entropy is measured in bits, so an entropy of 1 bit per output bit is ideal.

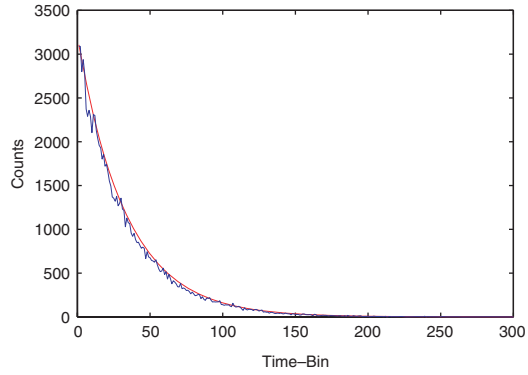

Figure 2. Actual (blue) vs. theoretical (red) waiting-time distributions of raw data given an average detection rate of 6 MHz and a time-bin resolution of 5 ns. Here we have chosen the first time bin to be *after* the 45-ns dead time of the detector. Deviation from the expected decaying exponential is due to systematic noise within the system. (The color version of this figure is included in the online version of the journal.)

For the distribution associated with our Poissonian light source, we assume the following simple model:

$$P_i = \begin{cases} 0 & \text{for } i < \delta, \\ \frac{e^{\frac{1}{\lambda}}}{\lambda} e^{-i/\lambda} & \text{for } i \geq \delta, \end{cases} \tag{2}$$

where δ is the dead time of the detector and λ is the average time between clicks, both measured in units of the counter resolution. For example, in an ideal system modeled by Equation (2) with $\lambda = 50$ and $\delta = 45$ ns, Equation (1) predicts an entropy of $\sim$7 bits per detection. In practice, we use the *measured* waiting-time distribution and Equation (1) to determine S. A more conservative option is to use the 'min-entropy' [10], which is the chance that an attacker will correctly guess an entire output string on the first try, given by $S_{\min} = -\log_2 \max(\{P_i\})$, where $\max(\{P_i\})$ is the maximum occurrence probability across all possible outputs of the hash function. For a given hash input block with 266 bits of entropy, the output min-entropy is approximately 255.28 bits per 256 bits of output, corresponding to a min-entropy per bit of 0.9972 [11].

3. Implementation

The implementation detailed here has four major components: photon production, optical attenuation, photon detection, and data processing and storage. Although our initial experiments used an LED light source, we now employ an attenuated laser diode, for two primary reasons. First, the light emitted from a typical LED is in principle in a thermal state (i.e. with photon bunching over a small interval), whereas laser emission is ideally in a coherent state; this advantage is

somewhat spurious, however, as the amount of bunching in an LED is miniscule and, in any event, our dead-time deletion automatically rejects such closely spaced photons. Second, it has been shown that a simple current-limited LED circuit can, under certain conditions, actually produce photon-number-squeezed light [12]. The resulting sub-Poissonian waiting-time distribution would display lower random fluctuations than that of a laser diode. As discussed below, the random deletion afforded by our strong attenuation should completely eliminate any such squeezing correlations. Nevertheless, to avoid all such concerns, we deemed it preferable to use the system with the simplest characteristics (the laser diode).

We have used a variety of techniques to attenuate the light to the desired photon flux. If we imagine that the photons are emitted in a perfect coherent state, governed by Poisson statistics, then in principle any random deletion process will not alter the statistics. In fact, even if the photons are produced, e.g., in a squeezed state [12], the correlations will be washed out by the large amount of attenuation required [13] – we typically operate with over 60 dB of optical attenuation, so that a given photon from the light source has less than a 10^{-6} chance of making it to the detector. Given this level of attenuation, numerical simulations show no noticeable difference in the amount of entropy per detection for both a coherent state and a heavily attenuated initially perfect number-squeezed state.

Since we rely on the independence of the photon arrival times, it is important that the physical process used to control the flux does not *introduce* unwanted (or unknown) correlations. For example, periodically gating the laser diode so that it is only operating for a short time would not be appropriate, even though the average flux might be as desired; in this case the entropy would be greatly reduced. We have implemented the necessary attenuation using three methods: spatial-mode selection (only collecting a small fraction of the emitted light), standard reflection neutral density filters, and a series of crossed polarizers. As predicted, in all cases we observed no significant difference in the final photon statistics, i.e. the waiting-time distribution was unaffected. Note that, in the case of polarization filtering, we are essentially relying on the same intrinsic quantum mechanical randomness that is assumed for many quantum cryptography implementations [1]; the security of the latter depends on the fact that a photon's transmission through a polarization analyzer is a truly random quantum event. Similarly, the reflective neutral density filters are an extreme limit of a simple beam splitter.

The transmitted photons are detected by a single-photon counter, in our case an avalanche photodiode (id Quantique 100-MMF50-ULN). Although this device's 45-ns dead time implies a saturation rate of over 22 MHz, in practice the device can only sustain a continuous count rate of 11 MHz, resulting in random number generation rates up to 39.8 MHz. We have also run successfully using a Perkin-Elmer avalanche photodiode (SPCM-AQR-13). In this case, however, we were further limited to a rate of 5 MHz (to avoid damaging the detector); after all processing the SPCM-based system reached random number generation rates up to 20.1 MHz.

The detector pulse is read by a field programmable gate array (Xilinx Spartan 3 FPGA) and input into a synchronous counter as the start signal. When a photon is registered, the counter resets to zero and then increments until the next detection event. A time-bin resolution of 5 ns is achieved by multiplying the internal crystal oscillator of 66 MHz to 200 MHz and treating each successive period as one time bin. Each time interval is assigned a value based on which time bin the photon was recorded in. The number of bits assigned to each detection is dependent on our expected entropy per detection, and consequently on the average detection rate (e.g. it would not be appropriate to assign more bits to each interval than the average rate of entropy allows). The data string for each detection event is truncated to this calculated length, and concatenated in a register until enough data is present to input into the hash function.

Because the probability distribution of the decaying exponential 'raw' counter data is not ideal, this data must be 'whitened' to remove bias and then compressed into the desired form. Cryptographic hash functions are commonly used for this purpose and we have chosen the SHA-256 algorithm [14] for our implementation. Unfortunately, as is the case with all hash functions, sometimes 'collisions' occur: cases where different inputs map to the same output. Because of this, we need to supply some extra entropy into the input buffer to make sure that each hashed output has approximately the same probability.

To quantify how much excess entropy is needed, we place two requirements on the hash function. First, every possible hash output should occur with the same frequency. Second, the bias in our raw counter input must not be present in the hashed output. For example, the smaller time bins will be more likely, and the highest-order bits of the counter data will typically be zero. To accommodate this we estimate the amount of entropy present and truncate any extra bits, only keeping the low bits – which are then input into the hash function. (For example, if the recorded interval for one particular detection pair is 11010111, and the waiting-time distribution corresponds to only an average of 5.5 bits of entropy per detection, then we

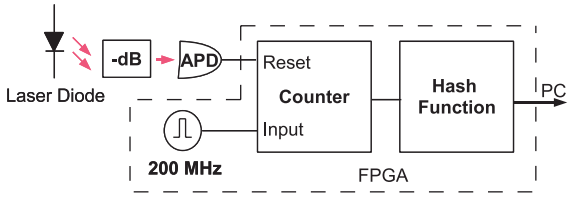

Figure 3. Data-flow diagram for our implementation. Photons emitted from the laser diode cause the APD to output pulses which are registered by the counter. For each new detection, depending on the time elapsed between subsequent detection events (as determined by the 200-MHz reference clock), the counter produces a random string. These counts are then accumulated until enough random data is present to 'whiten' using the SHA-256 hash function. The final data is then output through the PCI bus. (The color version of this figure is included in the online version of the journal.)

truncate this to 10111 before appending it to the string sent to the hash function.)

Given the uniform distribution assumption of our hash function, we can calculate the entropy of its output. If the hash function has Q possible outcomes, and the probability of each input is $P(X) \leq 2^{-T} = 1/N$ for some input entropy threshold T and time bin X, then the probability that m inputs map to a given output is given by the binomial distribution:

$$P = \binom{N}{m}\left(\frac{1}{Q}\right)^m\left(1-\frac{1}{Q}\right)^{N-m}. \quad (3)$$

By grouping all terms with the same P_i in Equation (1), it can be shown [11] that the resulting Shannon entropy is

$$S = \sum_m^N Q\binom{N}{m}\left(\frac{1}{Q}\right)^m\left(1-\frac{1}{Q}\right)^{N-m}\left(\frac{m}{N}\right)\log_2\left(\frac{m}{N}\right). \quad (4)$$

For our 256-bit hash function (which needs an input that has at least 256 bits of randomness), 10 extra bits of input entropy corresponds to an output entropy of 255.999, or an entropy per bit of 0.999996. One can also calculate the min-entropy, defined in the previous section.

Finally, the hashed data is output through the PCI port on the FPGA and stored on a desktop PC, where further manipulation and testing can occur. Figure 3 below shows the data flow of our implementation.

4. Performance of device/randomness tests

Our system outputs the data in two forms: the truncated 'raw' pre-hashed data, i.e. the concatenated string of the truncated low bits for each successive interval, corresponding to the waiting time between detections, and the 'whitened' hashed data. In our fastest configuration, the counts are output from the Avalanche Photo Diode (APD) at an average rate of 11 MHz, a speed easily handled by the FPGA. Each click produces 16 bits of data but, as discussed above, only the lowest ~8 bits are kept; this corresponds to 88 million bits of counter data per second. However, as discussed earlier, the average entropy of this truncated data is still not 1 random bit per output bit, but rather a value that depends on λ, the average number of time bins between detections and δ, the detector dead time. Our current rate of detection (11 MHz) and time-bin resolution (5 ns) correspond to an average of approximately 5.5 bits of entropy per detection, as determined using Equation (1) on a large sample of our data.

In practice, other effects cause deviations from the ideal exponential distribution, as shown in Figure 2. These include detector afterpulsing [15], noise and signal reflections, and timing effects within the data-processing electronics. (For example, in a previous implementation we determined which time bin a detection event occurred within using a ripple counter, for which later bits rely on the state of earlier bits. While having a speed advantage, this type of counter takes longer to transition between values when more bits of that value are changing, causing some time bins to be larger than others. Switching to a synchronous counter mostly resolved this problem.) These features are repeatable over different runs of the system, so by calculating the associated loss in randomness we can then perform extra hashing to correct for it. Therefore, in practice we use Equation (1) to directly calculate the actual entropy from the *measured* waiting-time distribution, allowing a small overhead to account for our statistical uncertainty in the actual distribution.

The truncated 'raw' data is input into the SHA-256 hash function in 432-bit blocks, a block size set by the hash function itself. Consequently, if each time interval is input as an 8-bit value, then each 432-bit block contains information pertaining to 54 detections. If each detection provides 5.5 bits of randomness, then this will give approximately 298 bits of entropy. The SHA hash function was designed to handle inputs of arbitrary size, with the general rule that the output entropy is approximately 10 bits less than that of the input. Therefore, in our system the entropy input to the hash function is comfortably above the required threshold of 266 bits. While slightly detracting from the net efficiency – we discard ~10% more random bits then strictly necessary – this provides added protection against fluctuations in count rate. For example, if the detections per second momentarily increase, the entropy per click will decrease; the extra input entropy provides a slight buffer against this. In a more advanced system, one could monitor the entropy generation rate in real time, adjusting the hashing

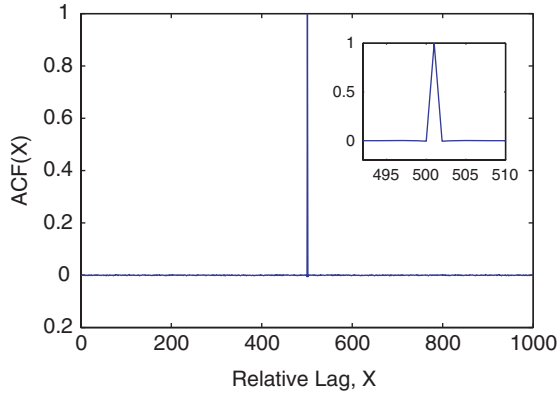

Figure 4. Autocorrelation function results from *pre-hashed* data for a correlation distance of 500. Inset displays the solitary peak in the middle, an indication of random white-noise behavior. (The color version of this figure is included in the online version of the journal.)

accordingly [16]. After all data collection and processing, our hash function outputs random data strings via the PCI bus at rates up to 39.8 million bits per second.

In order to test the amount of randomness present in our data, we have performed several tests, both on the raw data from the counter and again on the whitened data output from the hash function. Although there is no true test to determine if a sequence of bits is random, there are several widely accepted tests that we have utilized. The National Institute of Standards and Technology (NIST) has developed a comprehensive test suite designed to test the randomness of data from pseudo random number generators [17]. There are 16 categories of tests, including Fourier analysis, random pattern checking, binary matrix rank, and several others. Data was recorded to files with sizes ranging from 10 MB to over 200 MB (in 10-MB intervals) and from incoming detections at rates of 1 to 11 MHz (in 1-MHz intervals), and the tests were run on each of them. In all cases, the final whitened data passed every test in the suite. The 'raw' data obviously contains significant bias and failed a majority of the tests, but was still tested in order to look for correlations between events.

Additionally, to test how well the whitened hashed data fit the ideal output distribution, a series of chi-squared (χ^2) tests were run. The general χ^2 test, as well as the very similar 'overlapping serial test' [18], which looks for correlations within the data when paired into 'tuples', were performed; again, in all cases the whitened data was consistent with a random number string.

Unfortunately, simply passing these tests does not guarantee a perfectly random source, so it is critical to thoroughly understand the source itself. To detect any simple frequency patterns we performed an autocorrelation analysis on the raw pre-hashed data. The autocorrelation function is used in signal processing and statistics to measure how well a signal matches a shifted version of itself. The autocorrelation of a 'white-noise' signal – a signal of random noise with no frequency patterns – would look like a flat line except for a single sharp peak. Our raw data exhibited this behavior, as shown in Figure 4 below.

5. Discussion and future improvements

Presented above is a scheme and statistical validation for a cost-effective and fast quantum random number generator. Requiring only one photon detector and an inexpensive FPGA, this device could be reproduced and used as is for various scientific and commercial applications. However, there are several straightforward improvements that can be made.

Increasing the rate of incoming photons could increase the entropy of our final data, but there is a point where the entropy peaks and then quickly diminishes. In particular, as the count rate approaches the inverse of the dead time, the majority of the photons will fall into the lower-valued time bins, reducing the entropy per click. Taking into account the 45-ns dead time of our APD and assuming that our counter resets every time it receives a click, we are able to calculate the rate of entropy generation vs. the detection rate. As shown in Figure 5(*a*) below, given our current APD, the entropy peaks at around 13 million detected photons per second. However, this count rate is above the manufacturer's threshold of approximately 11 MHz for continual safe operation, where heating effects can destroy the device.

Because our detectors cannot handle continuous count rates higher than 11 MHz, we cannot fully validate the above theory with our current implementation. However, we tested the behavior using a similar implementation, one with lower time resolution and fewer clicks per second. The detection rate ('click rate') of a given APD with dead time DT and input photon rate R is modeled by

$$\text{Click rate} = \frac{1}{\text{DT} + \dfrac{1}{R}}. \tag{5}$$

Thus, given a set input rate, we can lower the maximum clicks per second by artificially increasing the dead time. In doing so, we were able to verify the predicted peak in entropy generation, as shown in Figure 5(*b*).

As illustrated in Figure 5(*a*), with our current time-bin resolution we are nearly at the peak rate

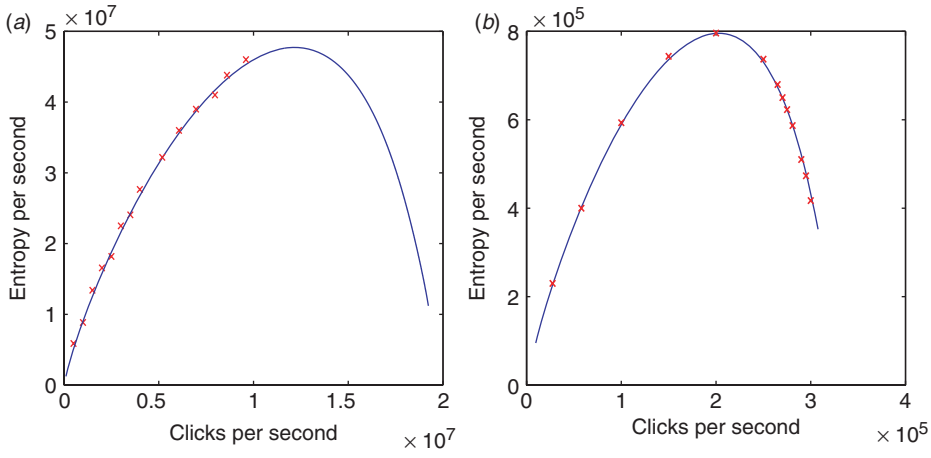

Figure 5. Theoretical and experimental entropy vs. detections per second, assuming (*a*) a detector with 50-ns dead time and 5-ns time-bin resolution and (*b*) a detector with 3-μs dead time and 300-ns time-bin resolution. (The color version of this figure is included in the online version of the journal.)

which maximizes the amount of entropy per second. Therefore, we have explored other ways to increase our random number generation rate. Although it performed very reliably during our experiment, the Spartan-3 FPGA is an older model. Upgrading the FPGA would have several advantages, including increased memory for possible parallel operations and the ability to synthesize a faster clock, thereby increasing our time-bin resolution and the achievable rate of random number generation. Additionally, we are currently exploring utilizing a time-interval analyzer chip, which would allow for 27-ps time bins, and could easily be integrated with our FPGA, thereby increasing the available amount of entropy per detection event, and predicting a net random number-generation rate of 90 MHz.

Along these same lines, we recently evaluated a high-speed time-interval analyzer (PicoHarp 300, by PicoQuant). This device offers a 4-ps time-bin resolution and allowed us to obtain approximately 17 bits of entropy per detection at the peak count rate of 10 MHz. Real-time hashing of the data has not yet been achieved, but post-processed data passed every set of tests that were run on the FPGA implementation, and would allow random number generation at a rate of approximately 110 MHz. However, in this case the time-bin resolution surpasses the jitter of the detector, and the randomness does not come from the light field alone, but also from the less well understood properties of the APD itself.

Because our implementation requires a set number of detections before hashing can commence, our QRNG has a latency of approximately 8 μs. For some applications (such as a loophole-free test of Bell's inequality [19]) which require 'on-demand' fast random numbers, this is a significant drawback. For others, however, such as quantum key distribution (QKD), the random numbers are used to determine the transmitted key bits and the receiver's measurement settings, which can be stored locally by the participants until they are needed. This assumes that the QKD system has secure storage; if this were not the case, QKD would be impossible anyway.

The issues listed above are currently the primary limiting factors in our design. We are confident that our electronics can handle higher input rates, and with the addition of alternative methods of photon detection we believe that output rates upwards of 1 GHz are achievable. For example, recently a silicon photomultiplier was shown to have a promising peak detection rate of 430 MHz [20], which when paired with higher time-bin resolution would greatly increase our random number generation rate. Also, multi-pixel photon counters – arrays of APDs packaged into one device – are currently being developed (Hamamatsu MPPC S10362-11); as well as offering an increased rate of random number generation by the techniques outlined above, it could be possible to gain additional *spatial* random information per photon by looking at *which* detector clicks at any given time.

Acknowledgements

We would like to thank Joseph Altepeter, David Ceperley, and Michael Neergard for helpful discussions. This work was funded in part by the DTO funded US Army Research Office Grant No. DAAD19-03-1-0282.

References

[1] Gisin, N.; Ribordy, G.; Tittel, W.; Zbinden, H. *Rev. Mod. Phys.* **2002,** *74,* 145–195.

[2] Stefanov, A.; Gisin, N.; Guinnard, L.; Zbinden, H. *J. Mod. Opt*. **2000,** *47,* 595–598.
[3] Jennewin, T.; Achleitner, U.; Weihs, G.; Weinfurter, H.; Zeilinger, A. *Rev. Sci. Instrum*. **2000,** *71,* 1675–1680.
[4] Wang, P.; Long, G.; Li, Y. *J. Appl. Phys*. **2006,** *100,* 056107-1–3.
[5] Lutkenhaus, N.; Cohen, J.; Lo, H. US Patent 7, 197, 523, 2007.
[6] Kwiat, P.; Jeffrey, E. Altepeter, P. US Patent Appl. 20060010182, 2006.
[7] Stipovec, M.; Rogina, B.M. *Rev. Sci. Instrum*. **2007,** *78,* 045104-1–7.
[8] Dultz, W.; Hildebrandt, E. PCT Patent WO. 98/16008, 1998.
[9] Rarity, J.G.; Owens, P.C.M.; Tapster, P.R. *J. Mod. Opt*. **1994,** *41,* 2435–2344.
[10] Renyi, A. In *On measures of information and entropy. Proceedings of the Fourth Berkeley Symposium on Mathematics, Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, Berkeley, California, June 20– July 30, 1962; Jerzy Neyman Ed.; University of California Press: Berkeley, CA, 1961; pp 547–561.
[11] Jeffrey, E. Ph.D. Thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2007.
[12] Tapster, P.R.; Rarity, J.G.; Satchell, J.S. *Europhys. Lett*. **1987,** *4,* 293–299.
[13] Bachor, H.A.; Ralph, T.C. *A Guide to Experiments in Quantum Optics;* Wiley: New York, 2004.
[14] Standard Hash Algorithm Specification, www.nist.gov/sha (accessed March 2008).
[15] Migdall, M.; Bienfang, J.; Polyakov, J. *Calibrating Photon-counting Detectors to High Accuracy: Background and Deadtime Isues;* NIST Special Publication: Gaithersburg, 2006.
[16] Neergard, M. *Private communication;* Oak Ridge National Laboratory: Oak Ridge, TN 37831, 2007.
[17] Rukhin, A.L.; Soto, J.; Nechvatal, J.R.; Smid, M.; Barker, E.B.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, N.A.; Dray, J.F.; Vo, S. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications;* NIST Special Publication: Gaithersburg, 2001.
[18] Xu, X.; Tsang, W.W. *AsiaSim* **2007,** *5,* 298–306.
[19] Weihs, G.; Jennewein, T.; Simon, C.; Weinfurter, H.; Zeilinger, A. *Phys. Rev. Lett*. **1998,** *80,* 5039–5044.
[20] Eraerds, P.; Legre, M.; Rochas, A.; Zbinden, H.; Gisin, N. *Opt. Express* **2007,** *15,* 14539–14549.