True Random Number Generators

Chapter · November 2014

DOI: 10.1007/978-3-319-10683-0_12

Mario Stipčević
Ruđer Bošković Institute

True Random Number Generators

Mario Stipčevič and Çetin Kaya Koç

Abstract Random numbers are needed in many areas: cryptography, Monte Carlo computation and simulation, industrial testing and labeling, hazard games, gambling, etc. Our assumption has been that random numbers cannot be computed; because computers operate in deterministic way, they cannot produce random numbers. Instead, random numbers are best obtained using physical (true) random number generators (TRNG), which operate by measuring a well controlled and specially prepared physical process. Randomness of a TRNG can be precisely, scientifically characterized and measured. Especially valuable are the information-theoretic provable RNGs, which, at the state of the art, seem to be possible only by exploiting randomness inherent to certain quantum systems. On the other hand, current industry standard dictates the use of RNGs based on free running oscillators (FRO) whose randomness is derived from electronics noise present in logic circuits and which cannot be strictly proven as uninformly random, but offer easier technological realization. The FRO approach is currently used in 3rd and 4th generation FPGA and ASIC hardware, unsuitable for realization of quantum RNGs. In this chapter we compare weak and strong aspects of the two approaches. Finally, we discuss several examples where use of a true RNG is critical and show how it can significantly improve security of cryptographic systems, and

Mario Stipčevič
Rudjer Bošković Institute
Zagreb, Croatia
&
University of California Santa Barbara
Santa Barbara, California 93106, USA
e-mail: stipcevi@gmail.com

Çetin Kaya Koç
University of California Santa Barbara
Santa Barbara, California 93106, USA
e-mail: koc@cs.ucsb.edu

discuss industrial and research challenges that prevent widespread use of TRNGs.

1 Introduction

True random numbers and physical nondeterministic random number generators (RNGs), seem to be of an ever increasing importance. Random numbers are essential in cryptography (mathematical, stochastic and quantum), Monte Carlo calculations, numerical simulations, statistical research, randomized algorithms, lottery etc. Today, true random numbers are most critically required in cryptography and its numerous applications to our everyday life: mobile communications, e-mail access, online payments, cashless payments, ATMs, e-banking, internet trade, point of sale, prepaid cards, wireless keys, general cyber-security, distributed power grid security (SCADA) etc.

Without loss of generality in the rest of the article we will assume that generators produce random bits.

In applications where provability is essential, randomness sources (if involved) must also be provably random otherwise the whole chain of proofs collapses. In cryptography, where due to the Kerhoff's principle all parts of protocols are publicly known except some secret (the key or other information) known only to the sender and the recipient, it is clear that secret must not be calculable by an eavesdropper i.e. it must be random. For example the well-known BB84 quantum key distribution protocol [4] (described in Section 3.4) would be completely insecure if only an eavesdropper could calculate (or predict) either Alice's random numbers or Bob's random numbers or both. From analysis of the secret key rate presented therein it is obvious that any predictability of random numbers by the eavesdropper would leak relevant information to him, thus diminishing the effective key rate. It is intriguing [94] that in the case that the eavesdropper could calculate the numbers exactly; the cryptographic potential of the BB84 protocol would be zero. Indeed one of the recent successful attacks on quantum cryptography exploits possibility to control local QRNGs exploiting a design flaw of two commercial quantum cryptographic systems and one practical scientific system. This example, discussed below, shows that the local random number generators assumed in BB84 are essential for its security and may not be exempt from the security proof.

Lottery is yet another serious business where random numbers are essential. Due to the large sum of money involved (estimated 6 billion USD annually only online and only in the US [40]) some countries have set explicit requirements for random number generators for use in online gambling and lottery machines and have set certificate issuing authorities. For example, the Lotteries and Gaming Authority (LGA) of Malta has prescribed a list of requirements for RNGs, stipulated in the Remote Gaming Regulations act

[51]. A RNG that does not conform to this act may not be legally used for gambling business. These rules have been put forward in order to ensure fair game by providers and to prevent possibility that gamers manipulate the system by foreseeing outcomes.

Random number generators have been an occupation of scientists and inventors for a long time. Whole branches of mathematics have been invented out of a need to understand random numbers and way to obtain them. In early seventies, at the dawn of modern computing era, John von Neumann was one the first to note that deterministic Turing computers are not able to produce true random numbers and put it in his well-known statement that "Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin".

Random number generators are one of the hottest topics of research in the last decade. There have been about 83 patents per year in the last decade, 1418 in total since 1970 and countless scientific articles published regarding true random number generators. Still, a sharp discrepancy between number of publications and very modest number of products (only 4 quantum RNGs and a handful of Zener noise based mostly phased-out RNGs) that ever made it to the market [37, 38, 71, 68] clearly indicates immaturity of most of the art. In our view main problems are lack of randomness proofs and poor reproducibility of majority of solutions presented so far. The search for true randomness continues.

2 Pseudorandom Number Generators

Historically, there have been two approaches to random number generation: algorithmic (pseudorandom) and by a physical process (nondeterministic).

Pseudorandom number generators (PRNG) are well known in the art and we are not going to address them here in great detail. Surveys and individual examples of PRNGs can be found elsewhere [45, 109, 36, 55, 57]. In a nutshell, a PRNG is nothing more than a mathematical formula, which produces deterministic, periodic sequence of numbers, which is completely determined by the initial state called seed. By definition such generators are not provably random. In practice, PRNGs feature perfect balance between 0's and 1's (zero bias) but also strong long-range correlations, which undermine cryptographic strength and can show up as unexpected errors in Monte Carlo calculations and modeling.

While most modern PRNGs pass all known statistical tests, there are myths about some PRNGs being much better than the others. The truth is that every PRNG shows its weakness in some particular application, Indeed PRNGs are often found to be the cause of erroneous stochastic simulations and calculations [66, 69, 11, 45, 51, 12, 58, 23, 32, 85, 104]. As for cryptographic purposes, all major families of PRNGs have been cryptanalyzed so

far [45, 89, 72] and use of PRNG where a RNG should be used will therefore present a big security risk for the protocol in question. We will revisit this point in more detail in Section 6.

In any case, due to strict determinism of PRNG algorithms, no PRNG is random by any reasonable definition of randomness. Let us illustrate this by a fictitious anecdote. Alice wanted to impress Bob, by a particular version of Mersenne Twister PRNG [57] for which she claimed that it produces true random numbers, by asking him to test them. Bob agreed but asked a minimum of 1 Giga bytes of random data to be sent to him via e-mail. Alice produced the huge file but her mailing program refused to send such a big file. Cutting a file into small pieces and sending multiple e-mails etc. was an option but too big a nuisance for both of them. Finally, Bob received from Alice a 1 kilo byte e-mail containing the following short notice: "Dear Bob, Please find attached a program in C++. Compile it, use the following seed: 12345678 and stop the program after producing 1 Giga bytes of data. That is what I wanted to send you". Instead of re-producing the file and running on his computer very time consuming tests, Bob shortly answered: "Dear Alice, if you think that 1 Giga bytes of truly random data can, under any circumstances, can be compressed without loss to just 1000 bytes, than I have nothing more to say to you!"

Advantages of PRNGs are their low cost, ease of implementation and user friendliness, especially in a CPU-available environment such as a PC computer but one has to be cautious when it comes to use of PR numbers for simulations, cryptography and in fact any use.

3 True Random Number Generators

Due to the Kerchoff's principle, the definition of a random number generator suitable for cryptography must include that even if every detail is known about the generator (schematic, algorithms etc.) it still must produce totally unpredictable bits. In contrast to PRNGs, physical (true, hardware) random number generators extract randomness form physical processes that behave in a fundamentally nondeterministic way which makes them better candidates for true random number generation. A physical RNG is a piece of hardware separate from the computer, usually connected to it via USB or PCI bus. Importing random numbers into a user program is complicated and requires original drivers. Prices range from 1k USD to 30k USD for bit production rates from 4 to 150 Mega bits per second [37, 38, 71]. Examples of physical processes used to generate randomness include: Johnson's noise [64], Zener noise [92], radioactive decay [24, 29], photon path splitting at the two-way beam splitter, photon arrival times etc. [74, 92, 95, 24, 29, 110, 9, 107, 106, 13, 105, 42, 39, 26]. Unlike the PRNGs, physical random number generators suffer from uneven probabilities of zeros and ones, that is bias (b), defined as

the difference of probabilities of 1's and 0's:

$$b = \frac{p(1) - p(0)}{2} \tag{1}$$

and short-range correlations which are best captured by serial autocorrelation coefficients (ak), defined for example in [45]:

$$a_k = \frac{\sum_{i=1}^{N-k}(b_i - \bar{b})(b_{i+k} - \bar{b})}{\sum_{i=1}^{N-k}(b_i - \bar{b})^2} \tag{2}$$

where $\{b_1, b_2, \ldots, b_N\}$ in as N bits long random string and k is the lag or "order" of the coefficient. Both b and a_k are normalized such that they can take on values in the interval $[-1, 1]$ and that an ideal RNG exhibits $b = 0$ and $a_k = 0$. True RNGs are generally constructed such that the correlation among bits is small – which is namely the idea of randomness. In some cases the physical system that is measured is being "reset" to an initial condition after production of each bit in order to reduce auto correlation. Therefore in most cases only a few lowest order autocorrelation coefficients are significant, ideally only the first one, which is named autocorrelation and denoted by a.

There are very many constructions or true RNGs and research is still getting impetus, but in our view one can roughly classify the present art in four families:

- noise based RNGs;
- free running oscillator RNGs;
- chaos RNGs;
- quantum RNGs.

The tree of RNGs is illustrated in Figure 1. Mathematical, pseudo random generators can also be divided into several categories depending on type of the algorithm used.

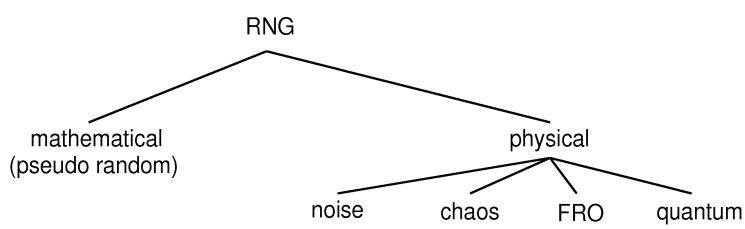

Figure 1: Classification of random number generators.

Note that our definition of a true RNG is not to be confused with a pseudo-random number generator implemented in CMOS logic or similar hardware; such generator is still a PRNG, since it is just an hardware implementation

of a mathematical method. Next, we are going to address each of the above families in some detail.

3.1 Noise-Based RNGs

Johnson's effect [64] creates random voltage on terminals of any resistive material which is held at a temperature higher than absolute zero. Johnson's noise is due to random thermal motion of the quantized electric charge (i.e. carriers). However, long-range carrier correlations in conductors cause correlations in movements of electric charges and therefore the resulting voltage is not completely random [3].

Zener noise (in semiconductor Zener diodes) is caused by tunneling of carriers through quantum barrier of ideally constant height and width. If current is sufficiently low, individual "jumps" of carriers through barrier will be seen as voltage peaks across the diode forming a pink noise of perfect randomness. An interesting property of this kind of noise is that at sufficiently high inverse voltage the diode exhibits high internal avalanche gain. Such gain mechanism leads to large amplitude of the noise and is highly insensitive to electromagnetic radiation from the environment. However, Zener effect is never found well isolated in physical devices from other effects nor is the quantum barrier constant. Most of the fore mentioned processes in resistors and Zener diodes have some memory effect. This means that an instantaneous voltage across the device depends on voltages in the (near) past and this in turn leads to a correlation among random numbers extracted there from.

Other popular sources of noise include: inverse base-emitter breakdown in bipolar transistors, laser phase noise [33], chaos noise [50] etc. The biggest problem with all kinds of noises is that randomness of noise sources cannot be well characterized, measured or even controlled during fabrication of the device. Furthermore, some noise mechanisms (notably Johnson's noise) produce rather tiny voltages that need to be strongly amplified before conversion to digital form. The strong amplification introduces further deviations from randomness due to the limited amplifier bandwidth and gain non-linearity. Also, fast electrical switching of binary logic used in the RNG circuitry produce strong electromagnetic interference so that multiple nearby RNGs (especially if on chip) tend to mutually synchronize causing the dramatic drop of overall entropy. On top of that highly sensitive amplifiers allow easy manipulation of noise-based RNGs by external electromagnetic fields which can be exploited for cryptographic attacks.

The general idea of noise-based true RNG is the following. The random analog voltage is sampled periodically and compared to a certain pre-defined threshold: if higher then "1" is generated, otherwise "0" is generated (Figure 2). It is obvious that threshold can be set so that the probabilities of 1's and 0's are roughly the same. However, fine tuning of the threshold poses

an insurmountable time-consuming problem and can never be done properly. For example, if tuning of bias to value of 0.1 requires 10 seconds, then tuning to 10 times lower value (0.01) would take 100 times longer (the required time scales as square of improvement ratio). And then there is a problem of stability: even the smallest drift of the mean value (for example due to temperature or supply voltage change) will create a large bias. Provability of any noise RNG is complicated and eventually made impossible by three reasons:

1. provability of randomness of the exploited noise source;
2. effect of the sampling/digitizing procedure;
3. eventual use of deterministic post-processing.

Going from this basic circuit, researchers have proposed many circuits whose aim is to improve the randomness, notably the bias.

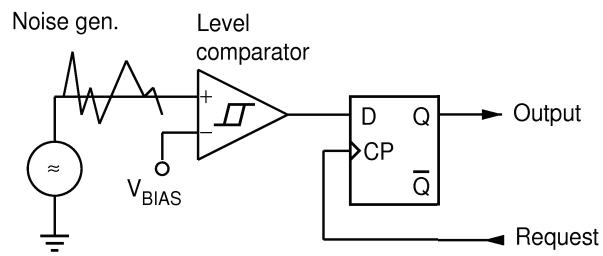

Figure 2: Noise-based RNG. Noise is fed to a level comparator whose output is either 0 or 1 depending whether its positive input input is below or above the threshold value VBIAS. Upon Request, fresh new random bit will sit on the Output.

First, the most obvious improvement would be to somehow de-bias the raw noise stream in hardware without the need of any adjustment of the threshold voltage. An interesting solution to that has been discovered by C. H. Vincent in 1970 [110], generalized by Chevalier & Menard in 1974 [9] and independently re-discovered later by Bagini and Bucci [1] and Stipcevic [92].

In the Bagini-Bucci generator [1] shown in Figure 3, analogue voltage from the free-running noise source is periodically sampled at frequency f_{Ck1} and compared to a threshold value at the Comparator. Whenever the comparator produced logical "1" the T-type flip-flop TFF changes its state. If the sampled process is random and stationary, because of time symmetry of this process the output of the TFF spends half of time in the low state and the other half in the high state. There are a couple of problems with that design. First, the holding capacitor acts as a memory that remembers previous analog voltage. Due to finite impedances in the circuit when charged with the next voltage level the voltage will be to some extent dependent on the

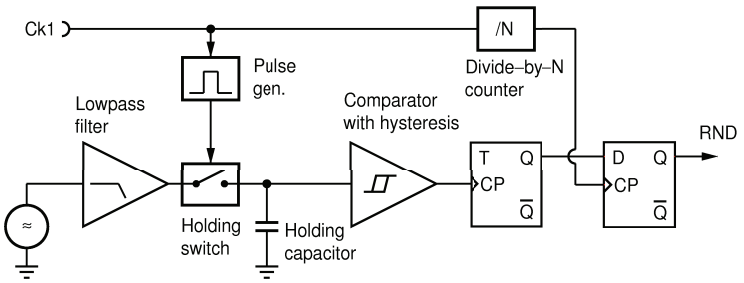

Figure 3: A zero-bias noise based RNG by Bagini and Bucci. The biased output produced by imperfect threshold principle is divided by 2 by a T flip-flop. The output of the T flip-flop spends exactly 50% of the time in state 1 and is sampled periodically by a pulse generator. The idea is that when sampled (by the D flip-flop) it will yield either 0 or 1 with perfectly equal probabilities. However, in practice non-negligible deviation from perfect bias will occur and correlations will exist.

previous one thus creating the auto correlation. The second problem with is that if the TFF is interrogated at too thigh rate it will tend to give the same answer several times in the row thus producing positively auto correlated output, even when the basic random process is truly random! The only way to circumvent this problem is to use bit sampling frequency f_{Ck2} much lower than the noise sampling frequency f_{Ck1}, for example $f_{Ck2} = f_{Ck1}/N$ thus arriving to asymptotically random sequence of bits in the limit of $N \to \infty$.

In the variation of this principle named "time summation of a random signal" [92, 91] shown ib Figure 4, time-wise random pulses at the output of the comparator COMP are counted by modulo 2 counter (TFF) whose output gets sampled upon a request send over the Request input. The results are similar to the Bagini-Bucci circuit except that bits can be generator faster because both the low pass filter and sampling circuits are not needed. Also, it features a naturally incorporated automated zero-bias loop consisting f the comparator COMP, low-pas filter with time constant much bigger than the bit sampling rate and amplifier OPA. The loop sets the threshold for the comparator in such a way that comparator spends half of the time in state "1" which is important to minimize autocorrelation. The TFF then takes care of complete canceling of the bias. In case of periodic bit sampling, again, correlation among bits will be non-vanishing even if the pulses are completely random unless the ratio between mean frequency of random pulses at the sampling frequency (N) goes to infinity. In practice however N only needs to be sufficiently large to keep correlations at the desired level.

The bad side of this "sampling" principle illustrated in Figures 3 and 4, is that it required N random events to produce one random bit (low efficiency). The good side is that by letting N be large enough, one can obtain any desired level of randomness quality, at least theoretically. Practically however, small imperfections in logical circuits will ultimately limit the achievable

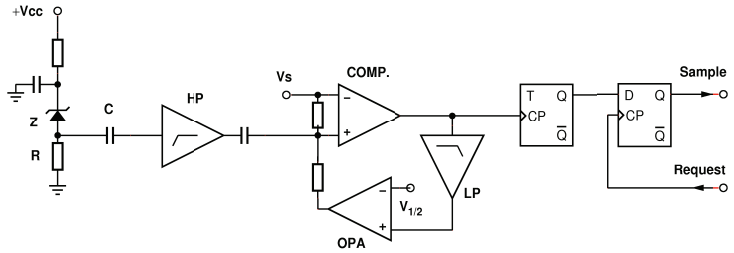

Figure 4: A zero-bias noise based RNG by Stipcevic. Time-wise random events appearing at COMP are summed at the input of the toggle flip-flop TFF and when sum becomes bigger than predefined time interval bit sampling period T random output is equal to number of random pulses in that interval mod 2. This is similar to Bagini-Bucci generator except that there is no need for low-pass filter and sampling circuit. There is no requirement that bits be sampled periodically. On top of that there is a automated zero-bias loop.

randomness. Regarding provability of randomness of this principle, technical imperfections of the individual components, unclear theory of operation of the "noise source", as well as overall complexity of the circuit makes it impossible to arrive to a credible proof of randomness.

Next example of noise class of RNGs is the Intel RNG [41] implemented in a limited series of computer processors (Figure 5). It uses amplified thermal noise of a resistor to disturb a voltage controlled oscillator thus arriving to a "slow" random pulse generator which is used to sample a "high speed" periodic oscillator. This fast-slow dichotomy is similar to the above described sampling RNGs and is known not to generate theoretically perfect randomness unless the ratio of fast to slow does tend to infinity. A particular peculiarity of this construction is that a voltage controlled oscillator (steady oscillator has zero entropy) is disturbed by a noisy voltage thus very probably yielding a lesser entropy than available from the noise source. The important property of such construction is that its frequency cannot surpass certain limit thus guaranteeing high enough ratio between fore mentioned high and low frequencies. It is therefore clear that the bits generated at the latch flip-flop (Super Latch) are not very random and require post-processing which consists of a modified (and patented) von Neumann method of efficiency 1/4 [100].

There is yet another Intel RNG appeared in 2011 after "10 years of research" which is apparently extremely simple [100] (Figure 6a). The idea is to obtain a circuit that does not have any (apparent) analog parts and is therefore compatible with logic chips. The circuit consists of two Yin-Yang connected inverters and two "oddly connected transistors". Authors explain that this circuit has two stable states: 0 and 1. If everything is perfectly symmetric, when transistors are driven high the output will end up in either low or high state. Authors further explain that even though ideally the output

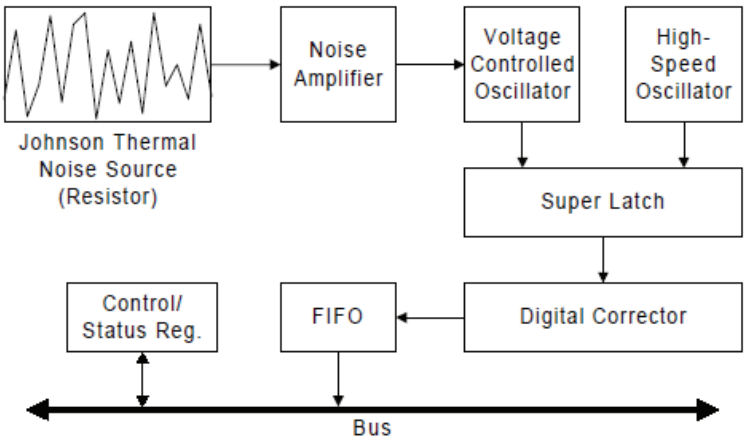

Figure 5: A Johnson noise based RNG by Intel. A high speed periodic digital oscillator is sampled at approximately random times defined by a Johnson noise signal. Time-wise random events appearing at COMP are summed at the input of the toggle flip-flop TFF.

value should be random, even the smallest difference in speed or strength of inverters would lead to high imbalance between zeros and ones (we would add: and possibly to complete lockup). Therefore Intel has put an additional current injecting mechanism that makes inverters controllable enough so that they can be made "equal". The quality of random numbers must be very low, because Intel uses 2 stage post processing in order to remove bias and correlations (Figure 6b). First stage is an unspecified randomness corrector after which "raw" bits become "high quality random seeds". The second stage is a PRNG seeded by these high quality seeds. It rests unclear why would high quality true random numbers be passing through a PRNG, but there might be only two reasons. Either these hardware numbers are not very good and must be further processed by the PRNG or/and Intel must comply with FIPS PUB-140 [21] which explicitly does not endorse any true RNG for cryptographic purposes and in this way numbers technically exit from a PRNG.

All above examples utilize electronic noise: a resource which is becoming less and less available because manufacturers of electronics components and chips make every possible effort to make it ever smaller. Therefore researches have turned to sources capable of producing fluctuating voltage similar to electronics noise but whose origin is more fundamental and therefore has less sensitivity to technological advances. For example very fast noise can be obtained by lasers. Lasers exhibit very fast fluctuations which can be detected by fast PIN or avalanche photo diodes thus producing wide-band electrical noise.

One such example is the phase noise of a single laser (Figure 7) invented by the CREAM group [33].

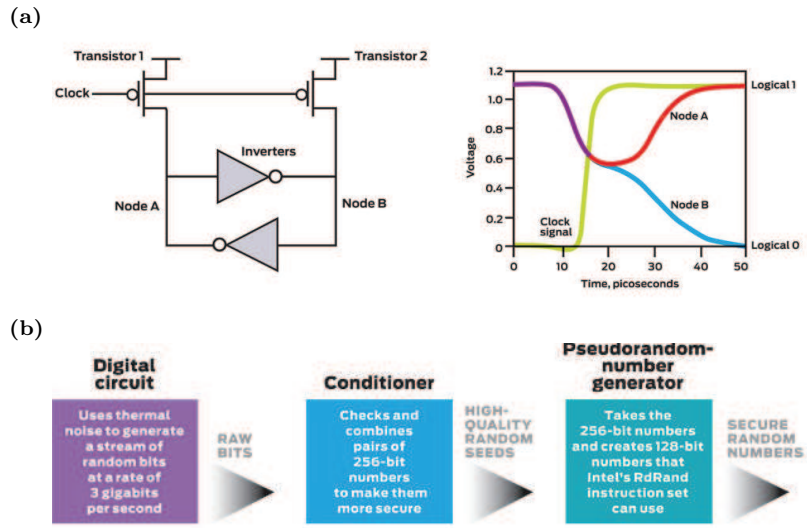

Figure 6: Intel's "quantum" random number generator. (a) Basic digital RNG circuit. Upon each pulse output stabilizes in random binary state. This is in fact yet another noise driven generator based on a specially prepared trimmed RS-type flip flop whose both set and Reset inputs are tied together and driven at the same time. The specialty of this flip-flop is that its inner gates may be current-trimmed in such a way as to make possible that output may stabilize in either low or high state. (Normally it would be locked to either state or produce high bias because of smallest asymmetry of its internal gates). (b) The postprocessing scheme.

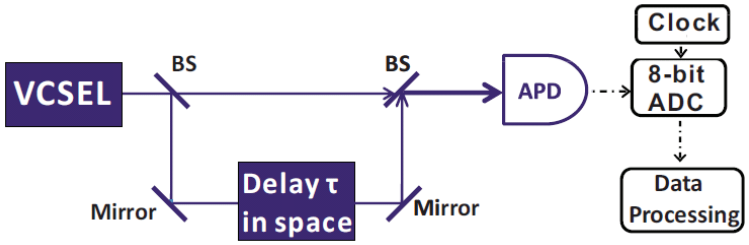

Figure 7: Laser phase noise based true RNG. Intensity of noise is determined by fundamental uncertainty of phase while its whiteness, that is Gaussian distribution of instantaneous amplitude, is due to the Central Limit Theorem.

This is an example of white noise based generator where Gaussian shaped distribution of analog electrical amplitudes has been obtained by optical means rather than electrical (for example such as discussed in Bagini-Bucci generator [1] and some others described above). The noise source, shown in Figure 7 is realized by use of a single mode VCSEL laser where the signal

and its delayed copy have been brought to interference on an APD detector via Michelson-Morley type interferometer, system also known as "homodyne" detection. The electrical noise produced at a very high gain-bandwidth photo diode is result of phase jitter of the laser. The noise voltage produced by the APD is then digitized by a fast (40MHz) analog to digital converter (ADC) with 8 bit resolution and the numbers so obtained are further processed to obtain random bits at 20 Mbit/sec. Authors show that if delay is much longer that the laser coherence time of 1.6 ns then the phase jitter is dominated by quantum effects which are separate from any construction detail and depend only on the laws of physics. In that regime, adding sufficient jitter leads to near-perfect Gaussian distribution via central limit theorem, similar to the principle utilized in [92]. Authors further measure the auto-correlation function of the analog noise and show that after about 10 ns all correlations die off. To be on the safe side, sampling of noise is made every 25 ns and after further simple post processing one obtained 20 Mbit/sec of random data that passed all relevant statistical tests (mentioned in Section E). Similar phase self-interference principle is exploited in [70]. The advantage of the quantum phase noise over the electronic noise is that its amplitude is determined by fundamental laws and is therefore (in the ideal case) independent of technological details of the laser. In our finding though, authors here were not considering two important points. First, the time delay introduces a "rolling" memory effect that necessary leads to auto-correlation of the noise voltage generated by the APD and therefore the bits obtained there from would not be random even if the phase jitter itself is random. Second, the bit generating algorithm, which most critically includes digitization of an analog quantum-random effect, is only approximate and a good care has to be exercised in order to keep randomness at the desired level at all times. Even so, this is one of very rare noise based generators which are characterized clean sequence of in-principle provable and well understood physical and algorithmic processes.

More examples of noise based true random number generators can be found in scientific literature and in the free-access world-wide patent database ESPACENET [22].

For all noise based generators some kind of post-processing is required. Simple ad hoc post processing like XORing several subsequent bits, von Neumann [63] de-biasing may be sufficient. But if the raw bits exhibit strong correlations, simple procedures may not be sufficient to eliminate correlations among bits which can even be enhanced by simple de-biasing procedures or changed from short-range ones to long-range ones. A better approach is found in complex, often offline post processing which however brings in its own problems (see Section 3.4).

There is a strong tendency among researchers to name noise based RNGs "quantum RNG" because noise is ultimately caused by small particles governed by laws of quantum mechanics. But noise is also a collective effect, a summation of many individual motions and therefore its quantumness is "blurred" by a collective behavior which is somewhere between quantum and

classical worlds. Furthermore, motion of particles which generate noise (for example electrons in a wire) is usually inter-correlated by action of forces among them to such extent that the noise may not be completely random [3]. Note that an autocorrelation of the order of a percent may not be important when motion of electrons is considered but if so generated random numbers with the serial autocorrelation of the same order (0.01) is used in numerical simulations, results may be completely wrong. Finally noise cannot be "restarted" in order to interrupt correlations between successive measurement/bit production.

In conclusion, a decent proof of randomness for present noise-based random number generators seems impossible because the underlying physical processes are not well isolated and do not rely on obvious or scientifically provable randomness.

3.2 Chaos RNG

Probably the most objectionable principle for physical generation of random numbers is to obtain them from repeated measurements of a physical system in chaos. Philosophical problem here is that chaos means finding order in what is seemingly random. So why would someone knowingly make use of a non-random system in order to generate random numbers? We are not aware of anyone so far asking or answering this question. In our opinion authors often resort to this type of generators because of three reasons:

1. conceptual mixing of chaos and randomness;
2. (mis)belief that hard-to-describe systems necessarily behave in random fashion;
3. robustness of certain chaotic systems to produce macroscopic levels of "noise" easily utilizable to generate random numbers essentially via noise RNG methods (as described in the Section A).

At present state of the art most convenient chaotic systems for fast generation of random numbers are optical, electrical or opt-electrical, although mechanical constructions have also been demonstrated, for example in [61]. In this section we present several typical designs.

Lasers can be brought to chaotic fluctuation of power by many different mechanisms. Well known are chaotic constructions involving distributed feedback lasers [50]. One very simple but extremely fast self-feedback chaotic laser system [42] is shown in Figure 8. Again, the light of chaotically fluctuating amplitude is detected by a fast photodiode (PD) whose amplitude is sampled by a fast 8-bit ADC and further processed by performing a high order differentiation, to yield a world record bit production speed of 300 GigaBit/sec.

Lasers offer means for realization of a very fast chaotic systems and are frequently used for random number generation. Due to a possibility to build

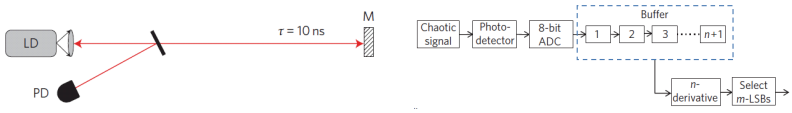

Figure 8: A chaotically behaved intensity of a self-feedback laser is read by a photo diode (PD) whose amplitude is sampled by a fast ADC and further processed by performing a high order differentiation, to yield a world record bit production speed of 300 GigaBit/sec.

tiny lasers, resonators and various passive and active optical elements on a chip, such generators could be completely integrated and could feature a low power consumption.

A random number generator shown in Figure 9 [50] consists of an ultra-wide-band (UWB) chaotic laser (a), amplitude sampler (b) and comparator (c). Its principle of operation is a copy-paste of the Bagini-Bucci noise generator described earlier (Figure 3) with the difference that instead of electrical noise here a light intensity of a chaotic laser is used as a source of randomness. The interesting distinguishing characteristic of this RNG is that it is "all optical", meaning that all signals and signal processing are done at the optical level, even the output numbers are in fact digital levels of light intensity: low light intensity signifies "0" while high intensity signifies "1". This is interesting for use in all-optical processing chips and furthermore, if so needed, the output can be easily converted into electrical signal by use of a fast photodiode and a suitable amplifier.

The UWB chaotic laser is made of two distributed feedback lasers, "master" and "slave" (Figure 9a) with master disturbing the feedback loop of the slave in such a way as to enhance bandwidth of it in chaotic regime [114]. The output intensity is extracted from the feedback loop by means of a beam splitter and sampled by an optical sampler at a constant sampling frequency determined by the mode-locked laser (Figure 9b). Each sampled value of light intensity is then compared to a threshold value by means of an all-optical comparator (Figure 9c) resulting in either high output intensity ("1") or low intensity ("0"). The random bits are produced at the pace of the mode-locked laser.

Bits so obtained are biased and somewhat auto-correlated. Since they at produced at periodic times, authors resort to a convenient bias and correlations reducing procedure by XORing simultaneous outputs bits of two identical, independent RNGs, as shown in Figure 10. Resulting random bits pass relevant statistical tests [50]. The chaotic behavior of the master-slave UWB laser has been theoretically modeled and bandwidth of the model shown to agree with experimental data [115], in an attempt to support claim of randomness of the above RNG. However, modeling or proving the shape and width of the noise spectrum of a source proves nothing about its randomness.

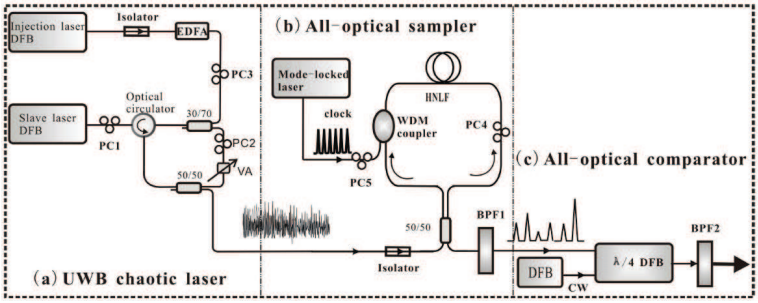

Figure 9: All-optical laser consisting of : a) Ultra Wide Band chaotic laser (UWB); b) All-optical sampler and c) All-optical comparator.

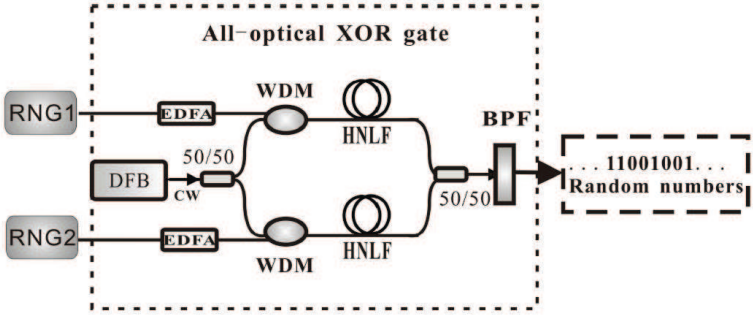

Figure 10: All-optical XORing of two independent RNGs reduces bias and correlations among bits.

In body of research related to chaotic RNGs, some authors claim to use system(s) in chaos without actually providing any direct evidence that the system in use for random number generation is indeed in chaos [102], some are able to demonstrate chaotic behavior for example by studying ballistic maps or Lyapunov exponent [73] and some even go so far as to model chaotic behavior of the system and confirm it experimentally [50, 114, 115]. But whichever the case, chaotic RNGs have a theoretic base common to those PRNGs which operate by simulating a deterministic chaotic system (for example) and therefore in the long run became short-breathed in producing new entropy, inevitably ending in producing not more than a small fraction of 1 bit of entropy per each new generated random bit.

General objection to the very idea of generation of random numbers by chaos is that chaotic behavior is defined as a specific type of solution of the differential equation which, supplemented by initial conditions, describes the system. Because any such equation and data contain only a limited (small) amount of information, once when that much information is extracted from the system by measurements there is no new information that can be ex-

tracted from it and consequently all further measurements contain (asymptotically) zero new information. In particular it means that a chaotic system, in theory, can only produce a limited set of random bits and that all the rest must be perfectly or near perfectly correlated to that set. Having said that, we understand that a realistic chaotic system never behaves exactly as it would by obeying the "equation of motion" that models it because of random quantum or statistical effects which randomize the system's phase-space trajectory all the time. However, these additional effects are not the basis for a chaotic RNG (and therefore not accounted for in its definition) and also are usually too tiny or ineffective to make any significant difference in a system whose behavior is mostly determined by a macroscopically observable chaos. On the other hand, fundamental quantum randomness alone can be harnessed for production of provable random numbers, as we will discuss in Section 3.4.

3.3 Free Running Oscillator RNGs

When output of a logical inverter circuit is fed to its input, the circuit turns to an oscillator, so called free running oscillator (FRO), Figure 11.

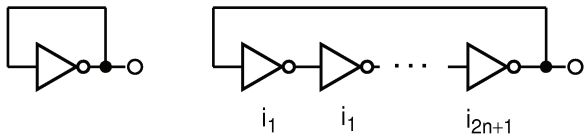

Figure 11: Schematic diagram of fast (left) and slow (right) FROs. Oscillation frequency is determined by internal delays and stray capacitances.

An inverting gate is in practice a very high gain inverting amplifier. Connecting its output to the input creates the Zeno paradox: if output is in logical HIGH state then the input will be as well and the NOT action will drive the output to go LOW. Once when the output goes LOW the NOT action will drive it to HIGH and so forth. Theoretical Boolean logic analysis will yield that the output is undetermined but in practice due to the finite propagation delay of the NOT element the circuit will oscillate. Peculiarity of this oscillation is that it appears in a circuit with negative feedback (180 deg phase shift) while in electronics theory negative feedback leads to "stabilization" rather than to oscillation. The reason for that is that by analyzing logical states we assumed infinite gain. However, since in practice gain is never infinite, it may happen that the circuit locks (stabilizes) into some voltage state between zero and one without any or with very small amplitude oscillations which are not capable of driving further logic circuits. To help oscillations

one may intentionally add some reactance in the feedback loop so to produce phase shift different from ±180 degrees. The same function may be provided with stray reactances. In that case the Barkhausen criterion may be satisfied for some high-frequency pole and oscillations will appear. Due to the complex mechanism of free oscillations, their frequency is typically quite sensitive to variation of power supply voltage and temperature but these changes are slow compared to the oscillation frequency. On the other hand the electronic noise present at the input ads up to the signal fed back from the output and after being strongly amplified causes very fast, random jitter of frequency and phase of oscillations. In that sense, FRO RNG can be regarded as a special case of a noise-based generator. Since noise of each such circuit is individual it is reasonable to assume that the multiple oscillators even when on the same chip have different frequencies and that their mutual phases walk off randomly in time. But when multiple such oscillators are close to each other (for example on a single chip) they tend to synchronize through electromagnetic interaction facilitated by high gain of FRO amplifiers. In effect, the immense gain of NOT gates required to amplify tiny electronic noise to a noticeable level also helps to pick up any other nearby interference. This effect known as "phased interlock" [64] may adversely affect the performance of the design and is a major problem inherent with FROs. Interlocked rings have waveforms that share (nearly) the same phase and this will lead to (near) pseudo-random operation. The same effect of high gain makes FROs vulnerable to attacks with external electromagnetic radiation.

Basic principle of random number generation with FRO's is that that output of a fast FRO (which can be either logical 0 or logical 1) is sampled by a slow FRO. This is an equivalent of abrupt stopping of a quickly turning wheel of fortune. Because the wheel spins so "fast" it appears stopped at a "random" position. In case of two FROs it is important that the relative phase jitter between the fast and the slow FRO is both random and large enough. Clearly, if there is no relative phase jitter the output will provide repetitive binary pattern. If the jitter is random but small, deviation from the repetitive pattern will be small as well leading to near pseudo-random behavior. If FRO's synchronize or at least partially synchronize a pattern with stochastic excursion (noise) would appear. Apart from that, another very important problem with FRO RNGs is that the output amplitude of a FRO depends on details of the stray reactances and delays in the circuit. As explained above, for a particular circuit it may well happen that output amplitude of a FRO is too small to drive the logic circuitry or that FRO locks in some state and stops oscillating. Schmidt action at the input can help minimizing this problem but at the expense of lowering the oscillation frequency and complicating the fabrication.

In spite of all these problems, current security standards [76] practically dictate use of RNGs based on free FROs. The NIST standard FIPS140-2 [21] says: "There are no FIPS Approved nondeterministic random number generatora". Consequently, the FRO approach, currently is used in 3rd and

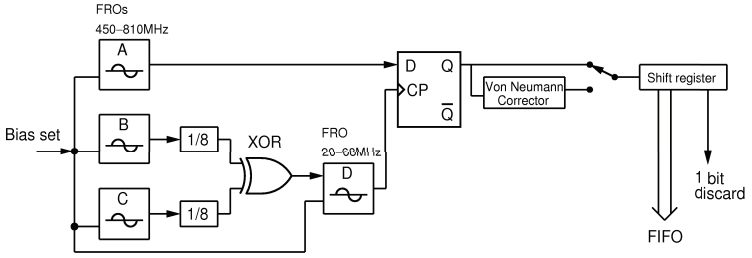

Figure 12: VIA C3 PadLock random number generator samples fast FRO (A) by slow FRO (D).

4th generation FPGA, CPLD and ASIC hardware for various cryptographic purposes. One real-life example that illustrates well the combinatorial cuisine typically needed to obtain a decent RNG is entropy source for PadLock "quantum" RNG implemented in VIA C3 processors [105, 106, 107, 108]. It consists of four FROs, 3 fast (450-810MHz) and 1 slow (20-68MHz). Wide tolerance on the frequencies already shows problems that we mentioned before: it is very hard to control parameters of FROs during fabrication. In this topology fast FRO (A) is sampled by a slow FRO (D) as discovered in the patent application [93]. At least one of the two FROs must be of good randomness and since it is easier to achieve with slower one VIA went for that option. The slow generator is made of FROs B, C and D. First, B and C are slowed down by 1/8 dividers and their XORed outputs are used to disturb slow FRO D (which is the only one featuring digital input). Resulting bits appear at the output Q of the D-type flip-flop in synchronization with pulses from the FRO D. Optionally, output is filtered through von Neumann corrector [63] which cuts the bit production rate roughly by a factor of 4 (see description in Section E). Looking at this schematic it is clear that it is impossible to arrive to a proof of its randomness. According to VIA [105], the analog bias voltage injected to this otherwise digital circuitry "may (or may not!) improve the statistical characteristics of the random bits". The bottom line is that the random numbers are still of low quality and in order to pass tests must be corrected (Section 3) by a full-blown secure hash algorithm SHA1 which is hardwired into the logic circuits on the same chip [105].

Because the digital logic chip infrastructure is unsuitable for realization of a quantum RNG (Subsection D), a FRO approach seem to be a reasonable viable alternative. However, caveat with FROs is that the semiconductor industry is making an enormous effort to make the electronics noise as small as possible and it generally goes down with newer versions of a chip. Consequently the effect of jitter can be very small and cause the FRO based RNG to operate in nearly PRNG regime. Therefore implementation details of a FRO based RNG most often have to be tailored for each specific type or

generation and technology of a programmable/reconfigurable or ASIC chip and uniformity of operation cannot be guaranteed from batch to batch.

A partial solution to above mentioned problems has been recently found in a novel synergistic combination of a linear feedback shift register (LSFR) [30] and FRO, called Fibonacci Ring Oscillator (FIRO) and Galois Ring Oscillator (GARO) [18]. The idea is to have a seeded LSFR-like PRNG which is realized as a clocked FRO. Such true random number generators does receive an initial state (seed) but although the seed sets the initial state, two identical generators with identical seed would diverge in time as they are under influence of (at least partially) individual noises. Figure 13 shows schematic of GARO and FIRO.

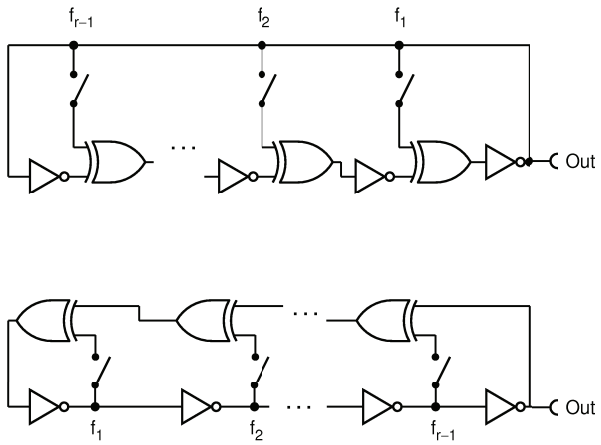

Figure 13: Galois Ring Oscillator (up) and Fibonacci Ring Oscillator (down). Number of stages defines order (r) while switches f_i define coefficients of the feedback polynomial.

Still, even with this interesting and innovative principle, the problem is cross-platform non-portability of the design and requirement of sufficiently large noise for the scheme to work in a reasonably random (far from pseudorandom) regime. Furthermore, the authors warn that design must be done most carefully in order to minimize interlocking with system clock and other logic circuits in the chip, including nearby FROs. Therefore they experimented with spatial placement of FROs in the chip. They also conclude that randomness of neither of the two generator families by itself is not perfect and could be "enhanced" by XORing two independent generators, most favorably one GARO and one FIRO.

More examples on FRO pre and post processing gymnastics, including XORing multiple generators, combinations with LSFRs etc. can be found in [96]. The complexity of post processing procedures required to pass the

statistical tests with FRO based RNGs is often such that any randomness proof is impossible, but even more interesting authors almost never seem to be aware that a proof is needed. A rare exemption in that respect is the work of Sunar et. al. [98] where a theoretical model of a FRO based RNG has been presented, analyzed and proven but later criticized in [118] as non-realistic. We however fond the whole proof unsatisfactory because it is based on the McNeill's model of FRO which simply postulates that free oscillations occur as a non stationary random process without actually linking the postulate to reality, for example by means of laws of physics. An excellent further reading and summary of problems and cuisine used to minimize them is found in [118]. Further reading on FRO based RNGs is given in [96].

In conclusion, FRO based RNGs are low-cost, low-entropy solutions whose only good side is the fact that they can be easily implemented in conventional programmable or reconfigurable logical chips which are used in various cyber-security solutions, but they do not offer either very good or provable randomness.

3.4 Quantum RNGs

What is a Quantum Random Number Generator? Since we live in the world governed by the laws of quantum physics any true random number generator (for example a rolling dice, or a flipping coin) may be named "quantum". However, we want to reserve this name for only those generators which utilize a single intrinsically random quantum effect (realized as close as possible to its theoretical idealization) measured over and over again in order to produce random bits in such a way that between any two sets of measurements used to deduce random bits, system is reset to same initial conditions. It may seem strange that such a physical setup (generator) is even possible, namely that starting from exactly the same initial conditions and measured in exactly the same way it gives different results, but quantum physics allows it. In this section we describe and explain multiple examples described in scientific articles and patents.

It turns out that some things in Nature come in the smallest amounts known as quanta. For example the electron carries the smallest quantity of charge, e. Similarly, there is the smallest quantity of information, called *qubit*. A single quantum of light (photon) can be used as a carrier of one qubit, but there are many other examples and they are not limited only to elementary particles. Qubit can be thought of as a linear combination of two bit values: 0 and 1. When a certain type of measurement is performed on a qubit it will "project" to either pure 0 or pure 1 state in the basis in which measurement has been carried out. Very often photons are used in QRNGs because they are easy to create, manipulate and detect. To illustrate this let us consider circularly polarized photon entering a polarizing beam splitter

(PBS), Figure 14. The PBS decomposes polarization of incident light and sends linear horizontal component on one output port and linear vertical component to the other port.

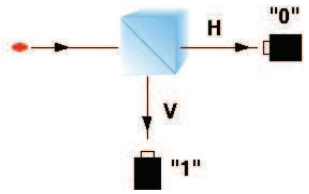

Figure 14: Spatial principle QRBG. Circularly polarized photon splits onto a linear horizontal/vertical analyzer with 50% chance to finish in either of the two output ports.

Thus, circularly polarized photon has equal content of both linear polarizations but since it cannot be split in half it has exactly 50% chance to exit either port. If now we label one of the ports as "0" and the other as "1" we immediately get a theoretically perfect RNG whose randomness is guaranteed by laws of quantum physics. Note that the system being "measured" is always the same yet it always gives a new random result. This is completely different from chaotic and noisy generators where in order to get a different result systems must change.

Quantum RNGs based on this (or other principles) can be made pretty good and the imperfections of any type (multi-photon emission, non-perfect circular polarization, beam splitter port axis misalignment, detector dead time, afterpulsing and memory effects, etc.) can be measured independently of the bit generation process so their effect on random numbers can be estimated with precision and dealt with post processing (see Section 3.5). This method is a basis for a commercial generator [37].

The main problem in practical realization of the beam splitting RNG is that it requires two detectors. Their initial differences and subsequent walk off with time due to aging and/or temperature effects will have an immediate impact on the quality of random numbers. For example if photon detection efficiencies of detectors are not perfectly equal, or if the beam splitter is not perfectly 50/50 percent, then the probability of ones will not be equal to probability of zeros. This problem can be minimized by use of a beam splitting scheme which utilizes only one photon detector [90] shown in Figure 15, but still the beam splitting ratio must be precisely adjusted mechanically. Leftover problems arise from detector dead time and afterpulsing leading to correlations which are impossible to eliminate completely but can be reduced below any desired level by targeted post processing.

The beam splitter RNG is an example of "spatial principle" in which value of the random bit, 0 or 1, is determined by the place at which photon ends

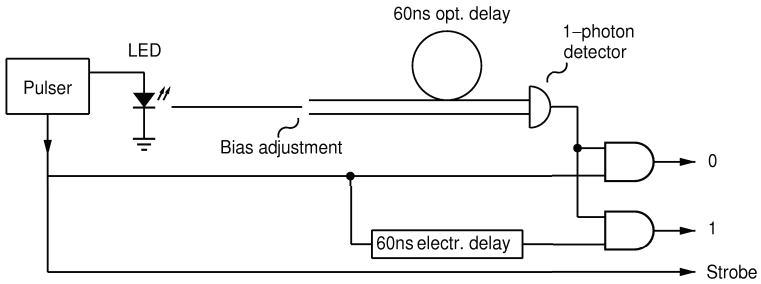

Figure 15: Optical quantum random number generator based on beam splitting which makes use of only one photon detector in order to avoid bias fluctuation with aging and initial tolerances.

up. A complementary "temporal principle" uses time information of random photon emission, for example in direct atomic (or quantum dot) relaxation, from well saturated lasers etc.

A simple time interval method shown in Figure 16, which is particularly immune to hardware imperfections has been proposed in [95]. It uses time rather than space information contained in random event generator (REG). In [95] photon emission and detection processes are used for the first time instead of much slower (and more dangerous!) radioactive decay [24, 29]. The bit production principle is as follows. Time intervals t_1 and t_2 spanned by three subsequent photon detections are compared: if $t_1 > t_2$ then produce "0", if $t_2 > t_1$ then produce "1", if $t_1 = t_2$ then produce nothing (skip).

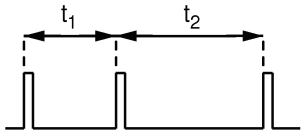

Figure 16: Timing principle QRBG. Photons from a single photon Poissonian source fall onto a single photon detector. Time intervals t_1 and t_2 spanned by three subsequent photon detections are compared: if $t_1 > t_2$ then produce "0", if $t_2 > t_1$ then produce "1", if $t_1 = t_2$ then produce nothing (skip).

The schematic of the physical setup is shown in Figure 17. Because only one photon detector is used, both bias and correlations are suppressed to almost undetectable levels yet there is nothing to be adjusted (unlike with the beam splitting principle).

Problem with this method is how the time intervals are measured. The crucial improvement made in [95] is the notion that clock measuring time intervals (ti) must be started in synchronization with beginning of each in-

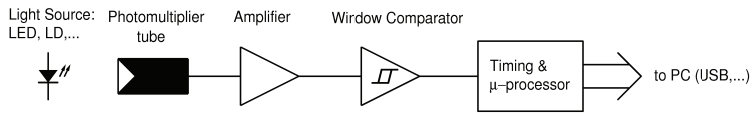

Figure 17: A general processing scheme of the temporal principle QRBG. Time-random photons fall onto the single photon detector consisting of a photomultiplier, amplifier and a comparator, such that each detected photon generates one logical pulse. Pulses are then processed according to the desired bit extraction principle and transmitted to a computer.

terval, otherwise the method would produce correlated bits even if fed by perfectly random events. This was not understood in previous works and patents [24] which consequently must have yielded correlated output but this was not detected at the time because clock frequency ($\approx$ 10 MHz) was much higher than the source mean frequency ($\approx$ 10kHz) in which case correlations are small. It can be shown that this method not only performs well at low ratio between clock and RPG frequencies but that it also cancels out almost all imperfections: intensity change of the source, efficiency change, dead time and afterpulsing of the photon detector. it is also highly immune to actual distribution of random interval times, as long as events are independent of each other. Furthermore, random bit production is self-clocked so if either source or detector die there will be no bits at the output. This generator was the first one found to be passing all known tests including "usual" t-statistical tests [54, 112, 78, 79] and some undisclosed algorithmic randomness tests [35].

A mixture of beam splitter and temporal principle is described in [39]. Unpolarized photon stream from the light source (LED diode) is passed through a polarizer reaching a polarizing beam splitter (NPBS), much the same as is the fore mentioned beam splitter RNG (Figure 14). With careful adjustment of the relative angle between polarizer and NPBS axis (ideally 45o) detectors D1 and D2 should produce random, mutually un correlated pulses of equal frequency (however adjustment of the polarizer angle is an insurmountable task, as explained for RNG in Figure 3. While pulses from D1 reset (inout R) the RS-type flip-flop setting the output to LOW state, the D2 set (input S) the flip-flop to HIGH state. The output of the said flip-flop is sampled at periodic times in order to generate a random bits.

Being combination of beam splitting and sampling principles this construction inherits the worst of the both:

1. bias is unstable (sensitive to temperature variations) and only mechanically adjustable;
2. correlations due to the finite sampling period as discussed in noise generators; and all that even if a perfectly random source of photons is assumed.

A commercial QRNG of Weinfurter et al. [26, 71] utilizing only the temporal principle is shown in Figure 19. Data-taking schematic is equivalent to

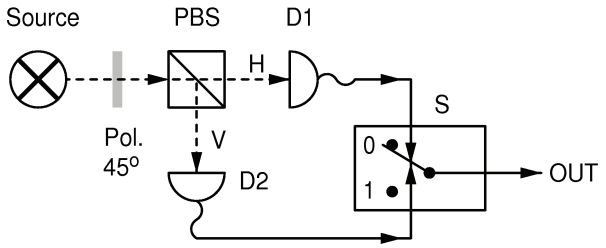

Figure 18: Optical quantum random number generator based on beam splitting and periodic sampling principles.

the general scheme given in Figure 17 with light source being low-intensity operated LED weakly coupled to a photomultiplier tube. Low coupling ensured low photon sampling rate on the order of 10^{-8} which suppresses any eventual photon correlations far beyond detectable level. The bit extraction method is implemented in FPGA reconfigurable chip and is as follows. Number of detected photons is counted in intervals of a constant time yielding a Poissonian statistics. Even number of events within an interval is interpreted as "1" and odd as "0". Authors note that due to non symmetric shape of the Poissonian distribution, probability of ones is not equal to probability of zeros. However, due to two imperfections in the photon detector (non zero dead time and dependence of dead time with the detection frequency) the resulting distribution is not Poissonian but more bell-shaped thus favorably leading to a bias that quickly tends to zero as the counting interval length rises. Authors show and compare experimental and theoretical results for modeled bias, however do not model or prove anything about correlations. Instead, correlations are simply evaluated from generated bits using linear autocorrelation coefficient. Theoretically, bias tends to zero as detection frequency goes to infinity. Empirically, the preferred operating condition is close to as high as possible detection frequency but a bit smaller due to rising problems in the photon detector. But in the same limit, it is to be expected that fluctuating bias produces increasing level of complex short-range correlations among bits – which however has not been mathematically modeled and/or brought into connection with the imperfections of the setup. The problem with this approach is that it fails to describe a theoretical model of a RNG that gives perfect random numbers based of an (nearly) ideally random quantum effect (for example low-intensity emission from LED) and assuming ideal apparatus. Consequently if fails to clearly prove randomness and to model deviation from perfect randomness introduced by implementation-related imperfections. Nevertheless, this generator has a practical value because it apparently passes all relevant statistical tests. It is however to be understood

that acceptable randomness proof cannot be obtained by passing any number of randomness tests (as will be discussed in Section 3.5)

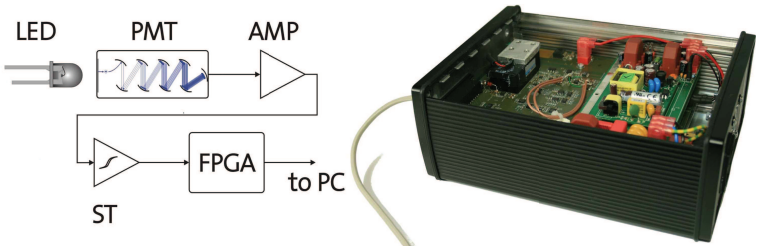

Figure 19: Optical quantum random number generator based on near-exponential statistics of photon time detection. The main detector imperfections, the dead time and pile-up, have been found to work in favor of smaller bias and serial autocorrelation which have been found to be as small as $2 \cdot 10^{-5}$ without post processing.

Yet another commercial quantum RNG which utilizes photon arrival time information has been presented by Picoquant [111, 68]. Here the complete chain of reasoning required for a convincing randomness proof has been at least attempted and, according to authors, successfully established. As in the previous example, a random event source of the type shown in Figure 17 is made utilizing essentially the same technique as in [26] (LED + photomultiplier tube). Specific difference of this construction with respect to previously described ones which use high speed photon detection and produce ≤ 1 bit per detection [90, 95, 39, 26], is that the random detections are made at relatively low mean frequency of 12.5 MHz thus operating in a regime far from dead time and pile-up effects producing a highly precise exponential distribution of time intervals (Figure 20 left). The time intervals $t_1, t_2, t_3, \ldots$ are measured by a nanosecond precision and quasi-exponentially distributed integer numbers so obtained are used to generate on average ≈ 14 random bits per each detected photon yielding ≈ 160 million raw random bits per second. The imperfections both in the extraction method and in hardware (timers, detectors, light source) are modeled resulting in a convincing lower bound on the average per-bit entropy of the raw bits. The average entropy is then improved by compression of the raw stream by resilient functions (see Section 3.5) to the level theoretically indistinguishable from true randomness even for bit strings of unrealistic length. The weakest link, in our opinion, is this last post processing step because it is not clearly proven that resilient functions are effective against the specific type of imperfections present in raw bits, that is that bounds on post processed bits hold. However, raw bits are already very close to random and further post processing by resilient functions clearly improves the pass rate of statistical tests indicating that the resulting bits are very close to true randomness. Indeed, post processed

bits at an average speed of 150 Mbit/sec pass all relevant statistical tests as well as some undisclosed statistical and algorithmic test performed by University of Twente research group [35]. Still, a caution is may be in order when resilient functions are used because some researchers [96] point out that resilient functions appear to be limited in their capability of eliminating the effects of active adversaries on the output bits.

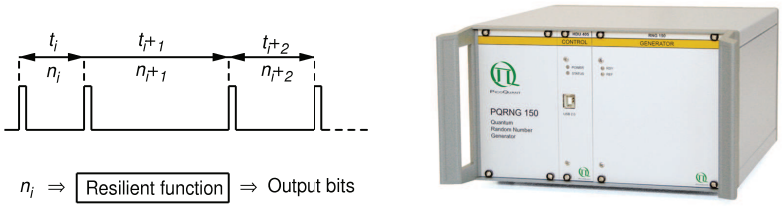

Figure 20: Optical quantum random number generator based on highly precise exponential distribution of photon detection times: schematic (left) product photo (right). The times between subsequent random events are measured by a very precise timing hardware resulting in integer numbers that represent the time. These numbers are then used to extract much more than 1 bit per detected photon resulting in 150 Mbit/sec overall average bit production rate obtained after a post-processing with resilient functions (see Section 3.5).

Yet an example of very fast (110 Mbit/sec) generator of the similar construction and philosophy as the previous one has been presented in [116, 117], Figure 21. In the first article, faint continuous light (from a LED) shines upon a photon detector which produces random events (detections) quite similar to the general system shown in Figure 17. Times between subsequent events are measured with a high resolution clock in order to obtain integer numbers that approximately follow exponential distribution. These numbers presented in binary form do not yield random bits because they have been drawn from a highly non-uniform distribution (namely, exponential cut off near zero at the dead time). In order to obtain more uniformly distributed numbers, in the subsequent article the light from the source (LED) is shaped in pulses with sharply rising power starting from the beginning to the end of each pulse. The idea is that by using carefully tailored pulse shape the times between subsequent photon detections would become uniform rather than exponential. There are caveats with this. First, the time intervals between photon detections are measured with a free running clock which has been noted in [95] to immediately lead to correlations even if incoming random events are truly random. Second, this scheme critically depends on the resulting distribution being exactly uniform while authors measured only approximate one. Third, by using a very high speed clock authors try to "squeeze out" as many random bits as they can from a single photon event (≈ 20 bits per detected photon) which generally leads to great amplification of hardware

imperfections thus leading to pretty bad raw random bits, as indeed was found. Fourth, the approximate results relating to the variable pulse power are both fundamental (i.e. pulse power should tend to infinity at the end of pulse proportional to $1/(t - t_0)$ where t_0 is the pulse length) and practical (pulse shape is achieved by an analog, only partially precise circuit thus not allowing to properly conduct proof of randomness. Authors also note that this circuit produces strong electrical disturbances in nearby circuitry which, in our view, makes it unsuitable for miniaturization to a chip level. And finally, the theoretical basis for exponential time-arrival distribution is drawn out of a steady field assumption whereas here strength of the light electromagnetic field is wildly varying so even theoretical grounds for this generator are not clean.

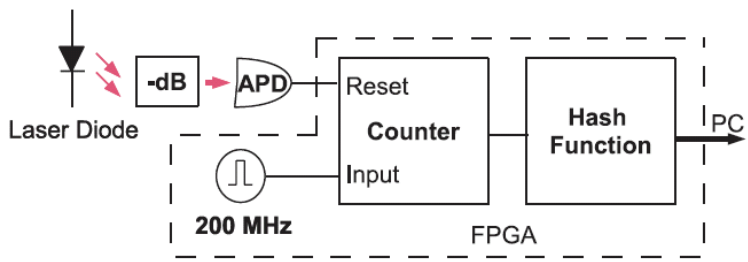

Figure 21: Optical quantum random number generator based on near-uniform photon arrival times from a specially shaped optical pulses.

This generator belongs to a broad niche of RNG constructs whose general philosophy is to produce partially random data and then filter it through a pseudo-random hash function (such as SHA256 used in this example) in hope to improve the randomness (see Section 3.5). We believe this is a very problematic approach and here is our reasoning. Proof of randomness in this case relies on estimating the entropy of the source of raw bits and on the process of randomness amplification by hashing. The hashing procedure is generally not foolproof [2] and does not allow just blind application of the hash function to a badly constructed generator. Let us imagine for example that raw RNG source produces some sequences more often then the others (which indeed is the case if it is non-random). Then the hash of these sequences (the hash function being deterministic) would also produce some sequences more often than the others meaning that even the "corrected" bits would not be random. A nice confirmation of this comes from this very example: even after hashing the produced bits are not completely random and fail some statistical tests.

We saw that photon emission and photon detection techniques are often used in quantum random number generators. Photon detection rate of current single photon detectors is a limiting factor in achievable random bit product rate especially for semiconductor avalanche photo diodes (APD). APDs are

small and convenient for single photon detection on a chip-scale however suffer from imperfections that are especially bad for random number generation and consequently rarely used for that purpose. The biggest problem are relatively long dead time (induced by requirement to quench avalanche between subsequent detections) and high afterpulsing rate (usually in range 1-10%). In order to advance on this, Toshiba has developed a special, so called "self differencing" approach [119] to readout of semiconductor avalanche photodiodes which promises significantly higher detection rates (lower dead dime) than usual active quenching method while suppressing afterpulses by effectively squaring the afterpulsing probability. This new technique has been used for random number generation by the same group of authors [20]. Namely, even though this method does not offer spectacular improvements in general because it inherently prefers operating the APD at low detection efficiency, but it is very well suited for use in random number generation because of its high gating speed and complete irrelevance of the photon detection efficiency for that application.

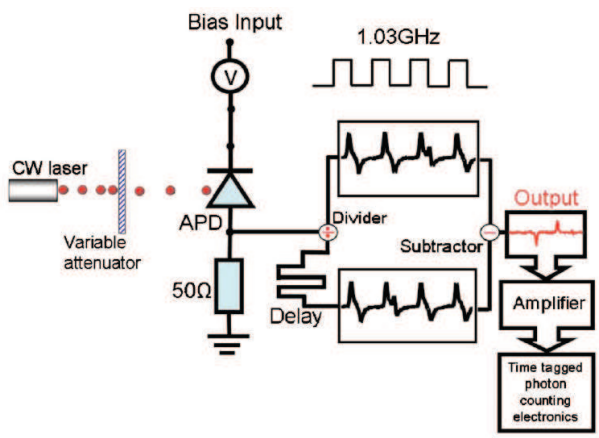

Figure 22: Optical quantum random number generator based on periodically gated, "self-differencing" operated avalanche photo diode (APD). The power of DFB cw laser (1550nm) is adjusted (by means of variable attenuator) such that strength of the electromagnetic field falling on the surface of the APD causes roughly 0.004 avalanche detections per gate, resulting in 4.01 MHz of random bits.

A distributed feedback laser (DFB) in continuous wave (cw) mode is used as the light source. The power of the DFB laser (1550nm) is adjusted (by means of variable attenuator) such that strength of the electromagnetic field reaching the surface of the APD causes roughly 0.004 avalanche detections per gate. When detection occurs new random bit is generated and its value is "0" if it occurred on even gate or "1" if on odd gate. Taking into account

the detection efficiency of 0.004 this method yields 4.01 MHz of random bits. This bit generating process is intrinsically biasless (probabilities of zeros and ones are equal) but (what is not noted by the authors of this article) there is an intrinsic negative autocorrelation which rises with detection efficiency. Namely, in the limiting case of efficiency 1 (one detection per gate) there would always be a "1" after "0" and vice versa thus producing a completely deterministic sequence $01010101\cdots$ which has autocorrelation equal to -1. Even though authors claim that this method of generating random numbers could, in principle, be extended to much higher rates by using higher laser power and detection rate of up to 100 MHz (efficiency of 0.100) it is clear that at that point the autocorrelation would amount approximately -0.1 and bits would not pas any randomness test.

There are numerous other variations of space and time principles that can be found in the scientific and patent literature.

In conclusion, the most distinctive characteristics of a quantum approach to random number generation is that, at last in principle, it makes possible establishing of a simple relation between randomness of numbers, the exploited physical process and implementation imperfections, thus offering a possibility for scientific proof of randomness. Careful practical realizations come sufficiently close to theoretical idealization and allow for an independent assessment of implementation imperfections effects of which can, if required, be dealt by information theoretic postprocessing (see Section 3.5). On top of that quantum random detection processes exist that are inherently highly insensitive to electromagnetic radiation (e.g., avalanche amplification in semiconductor photo diodes) thus offering immunity to side-channel manipulation by external fields. Because of all said, quantum RNGss are the best choice for true random number generation for cryptography and other application which critically require true random numbers. Most significant drawback of present solutions is that they make use of bulky physical objects and therefore can not be miniaturized to the chip level using present technologies. Furthermore, due to frequent use of photon detectors, QRNGs are typically very expensive and much slower that software PRNGs. Fortunately, nascent science and technology of optical chips offers a promising avenue for fast, miniature and affordable quantum RNGs and significant advances can be expected in this exciting field in near future.

3.5 Post Processing

True random number generators can never be made perfect and therefore some post processing is usually required. There exist plethora of post processing algorithms whose purpose is to eliminate imperfections present in "raw" random numbers produced by physical generators. A good review of

post processing methods is given in [96]. Here we will only categorize and shortly describe main principles

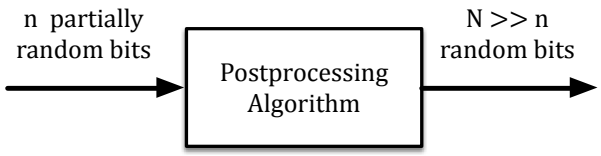

Figure 23: General schematic of random number post processing.

The general idea of post processing (Figure 23) is to sacrifice a certain percentage of bits in order to arrive to a smaller but more random set. There are basically four techniques:

1. ad hoc simple correctors;
2. whitening with cryptographic hash functions;
3. extractor algorithms [86, 87]; and
4. resilient functions [10, 80, 48, 49].

Although there is a "gray zone" of what part of random number production belongs to bit extraction method and which to post processing, the bit extraction is usually a first and very simple step which converts physical measurement of an analog or digital signal into the "raw" digital random binary number (such as digitizing analog noise via a threshold comparator shown in Figure 2), whereas post processing is a more complex process designed to reduce or completely move imperfections that are necessarily present after the first step. While bit extraction is always made in hardware, post processing algorithms are usually so complicated that they can only be executed by a computer (or a microcontroller or FPGA) although the most valuable post processing techniques are those simple enough to be suitable for direct implementation in hardware.

Generally, post processing takes a lot of resources and blurs In our opinion, a good true RNG should be post-processing free or use minimal ad-hoc postprocessing. Most popular post processing techniques can be categorized in four families as described in the following.

3.5.1 Ad hoc Simple Correctors

Ad hoc correctors examples are: XORing two or more neighboring bits from a same RNG [87], omitting bits (decimator), feeding a LSFR with imperfect random numbers [101], latin square bits reshuffling [54], von Neumann [63] and Peres [67] de-biasing, XORing two or more RNGs that work in parallel [50, 15] etc.

It is important to note that ad hoc, naive processing can lead to unexpected problems. For example it is usually considered a good idea to apply von Neumann de-biasing scheme [63] in order to completely remove any bias from the sequence of bits. The scheme works as follows. Biased bit sequence is cut into a sequence of non-overlapping pairs of bits. Pairs 11 and 00 are discarded, 01 is converted to "0" and 10 is converted into "1". While it is tempting to think that probability of occurrence of 10 is equal to probability of 01 (and therefore the resulting sequence has no bias), it is often overlooked that this this is true only if bits are completely independent (no correlations). The following extreme example illustrates how miserably von Neumann's procedure can fail. Let us consider the sequence: $101010101010\cdots$. It obviously readily has no bias. After application of von Neumann de-biasing the sequence reads: $111111\cdots$ which is a maximally biased sequence. The reason for this unexpected result is that the original sequence is maximally (anti-correlated) and therefore quite far from assumption of complete statistical independence. Generally, if the raw string is correlated, nave de-biasing procedure may even increase the bias or create other unexpected statistical deficiencies. On the other hand, simple and easy-to-understand ad-hoc correctors have the advantage, over more complex procedures, that they are easier to include in a randomness proof.

3.5.2 Cryptographic One-Way (Hash) Functions

One-way hash function is a mathematical function whose domain is a whole set of integer numbers and whose output is a binary number of exactly N bits, where N usually is in the range from 128 to 512. Hash functions are characterized by two requirements:

- Given an output value there is no faster way to find a corresponding input than by random guessing (that is a hash function is "one-way");
- The probability of two different inputs yielding the same output is less or equal to $1/2N$.

One of the most popular post processing techniques is "whitening" of output of a TRNG by means of a cryptographic hash function. such as MD5, SHA-1, SHA-2, SHA-256, SHA-512. Many authors believe that a bad RNG that does not pass statistical tests, run through a "cryptographic" hash compression procedure would magically become very good, without actually demonstrating any theoretical understanding on why this should be the case. Indeed a very interesting example given in [117] demonstrates that hashing a bad generator can fail to enhance randomness enough to pass statistical tests.

From a performance perspective, implementing a hash function in hardware chips is pretty resource-demanding so in most cases hashing is done on a compute; two exemptions to this rule are the aforementioned Intel's RNG in

Figure and VIA C3 in Figure 12, which make use of SHA-1 hardwired right next to the RNG on the same chip. Regarding the provability of randomness of the hashed output, even though interesting results on privacy (and randomness) amplification have been theoretically exercised for Wegman's Universal Hash Function(s) [6], in case of a real-life, black-box hash functions (which probably contain unknown statistical or security weaknesses) it is hard to perform a convincing proof of randomness. For example a hash function may contain statistical problems like for example some output strings being more probable than others which would then be inherited by the output bits even if the function is fed by perfect random numbers. On top of that hash functions are usually used at the end of the post processing leaving in mouth a bitter aftertaste that physically generated random numbers actually exit out of a deterministic, complex, black-box piece of software which has not been specifically designed for the purpose.

3.5.3 Extractor Functions

More scrutinized approach to randomness healing is offered by the young theory of extractors [86]. A randomness extractor is an algorithm that converts a long weakly random sequence into a shorter sequence with almost perfect randomness. For some randomness sources IT provable extractors exist but no single randomness extractor currently exists that has been proven to work when applied blindly to any type of a high-entropy source. Problem with extractor algorithms is that they require a memory buffer and a lot of CPU which slows down the overall output bit rate.

Extractor functions for post processing of true random number generators were proposed by Barak, Shaltiel and Tomer [2]. The initial purpose was to achieve designs robust against changes in the physical generators due to for example aging, temperature changes or attacks. Extractor functions are stateless functions with quantifiable properties originally developed as a tool for complexity theory. Fore mentioned group of authors has developed a mathematical model to capture an adversary's influence on the randomness source and give an explicit construction based on universal hash functions which is proven for its output properties even if non-local correlations exists in the input source.

More on the theory and practice of extractors can be found in [87].

3.5.4 Resilient Functions

Yet another approach to enhancing randomness by filtering through some deterministic process is the use of resilient functions that were introduced by Sunar, Martin, and Stinson in [98] as the post processing step for a FRO RNG design. The idea is, according to authors, to "filter out any deterministic bits"

from the raw string in the environment where some bits may be under control of an attacker and that bits are then considered "deterministic". Authors of [98] study degree of resilience of the procedure against active adversaries (there from comes the name of these functions). In short, an (n, m, k)-resilient function is a function $f : F^n \to F_m$ such that every possible output m-tuple is equally likely to occur when the values of k input bits are fixed and the remaining $n - k$ bits are each chosen at random. The elements of F are binary values 0 and 1. The important distinguishing characteristic of resilient functions is that they have been constructed specifically to nullify the attacks on (certain percentage of) random bits – a point of high importance in cryptographic applications of random numbers (see Sections 4-6).

More on the theory and practice resilient functions can be found in [10, 98, 80, 48, 49].

4 Randomness Evaluation (Testing)

The most important notion about statistical testing is the following: if a generator passes all known statistical tests this does not prove that it is random: it only means that it passes all currently known randomness tests. Tomorrow it can fail some new test or it already fails in the way known only to its constructors.

Most randomness tests check one or more statistical properties of long sequences of random numbers, for example bias, serial autocorrelation etc. Some compilations of tests are more oriented towards problems in PRNGs (eg. DIEHARD [54]) some more to true RNGs (e.g., ENT [112]) while some are of general nature (e.g., Universal Test [59], NIST STS [79]). The unfortunate fact is that there is an infinite number of statistical properties which truly random numbers must satisfy. Tests themselves are not perfect: some contain errors discovered latter [50, 76] or constants of questionable precision obtained by simulation using "trusted" random number generators such as combination of white noise and "black noise" [54].

Running a comprehensive set of tests takes many CPU hours: to test 1E9 bits with NIST STS it takes about 6 hours on the fastest single core CPU while to produce that much bits with a commercial QRNG it takes between 7 and 250 seconds.

Randomness tests are very time consuming – it takes much shorter time to generate numbers than to test them. Nevertheless, randomness testing is important for constructors of RNGs. Therefore in some cases where one can reasonably expect only certain type of imperfections (especially for quantum RNGs) one will tend to use only special tests sensitive to these particular imperfections in order to arrive to more efficient testing.

5 Random Numbers in Quantum Cryptography

Quantum cryptography is a protocol of public agreement of a symmetric cryptographic key, meaning if two parties A and B possess a small common secret key then using this protocol they will be able to establish a common secret key of any length. This cryptographic function is also known as "secret key growing". The ultimate goal of establishing a long secret key is to use it as a one-time pad and thus obtain transfer of data in absolute secrecy. There are several mathematically identical QC protocols. The first one, named after its creators as BB84, appeared in 1984 and has been experimentally realized in 1991 [4].

In the BB84 scenario, Alice and Bob are connected via two different channels: the quantum channel (usually well shielded optic fiber) capable of conducting single photons of light and an un-secured "classical" channel such as a telephone line, radio link or internet.

Here is the simplified schematic of how the protocol works: Alice can prepare photons in different polarization states. In order to establish a secret key, Alice sends to Bob a sequence of random numbers encoded in photon polarizations as follows: "1" is equiprobably encoded either as linear-vertical (LV) or left-circular (LC) polarization, while "0" equiprobably encoded either as linear-horizontal (LH) or right-circular (RC). Bob, has two polarization analyzers: one which can correctly measure linear polarizations (L) and the other which can correctly measure circular polarizations (C). Alice chooses one polarization at random, prepares the photon and sends it to Bob. Bob chooses one of the two analyzers at random and measures with it the photon received from Alice. If, by chance, Bob has chosen right polarizer he will receive 0 or 1 as sent by Alice. If Bob has chosen wrong polarizer he will receive 0 or 1 with equal probability regardless of what Alice has sent. So, after receiving a photon from Alice Bob announces (over authenticated but nit secret public channel) which polarizer he has just used (L or C). Note that this says nothing to potential eavesdropper about the value of the bit Bob has got. Alice responds with "Keep it" or "Trash it". So, bit by bit the two of them are building their secret key. Laws of quantum mechanics prevent qubit from being faithfully copied so an eavesdropper can obtain only a limited information about Alice's and Bob's string and furthermore eavesdropping can be detected by Alice and Bob.

It is straightforward to see that the whole protocol would be completely insecure if only the eavesdropper could calculate (or predict) either Alice's random numbers or Bob's random numbers or both. From analysis of the secret key rate presented in [5] it is obvious that any predictability of random numbers by the eavesdropper would leak relevant information to him, thus diminishing the effective key rate. It is intriguing (and obvious) that in the case that the eavesdropper could calculate the numbers exactly, the cryptographic potential of the BB84 protocol would be zero. This example shows

that the local random number generators assumed in BB84 are essential for its security and should not be taken for granted.

Apart from what has been described above, the BB84 protocol has two more subprotocols. Namely, due to the quantum incoherence, losses in the quantum channel or eavesdropping Alice and Bob will not have the exact same strings of bits after the first phase, although the two strings will have a lot of common information. Therefore the second subprotocol, the "data reconciliation", is used to equalize the two strings, albeit at a cost of leaking some small information to an eavesdropper. Fortunately, Alice and Bob can calculate a lower limit of their mutual information after the two initial phases and then perform the privacy amplification phase in order to arrive to a shorter but much more private key. These two subprotocols require further random numbers.

The protocol BB84 is considered information theoretically proven [88, 31] meaning that an attacker simply has no enough information to calculate the plaintext even given infinite computing resources. This is in strong contrast with widely used "deterministic cryptography" where an attacker has enough information to calculate the key except that it would probably require insurmountably large computation resources and/or time. The caveat with QC is that the security proof holds only against family of attacks considered in the proof. Unfortunately, with time, it became evident that unexpected attacks on QC which utilize various quantum effects are feasible which makes QC much less "untouchable".

For example in 2007 an MIT group presented attack that that gave Eve as much as 100% of information about the key albeit at an expense of elevated BER [44], but the attack was reassuringly classified as "simulation only" because it assumed that Eve has a specific information about Bob's receiver that she apparently could not get.

As with any other cryptographic procedure, some problems in real-world implementation of the protocol, especially of the quantum channel and real photon detectors could be used to weaken the cryptographic security of the protocol and open pathways for attacks.

A beautiful demonstration of serious weakening and even 100 percent breaking of the key without any notice to legitimate parties has been made by Makarov et al. in 2010 [52, 27]. The demonstration has been made on the commercial QC systems from Swiss company IdQuantique, based in Geneva, Switzerland, and one by MagiQ Technologies, based in Boston, Massachusetts. Improvements that would make QC resistant to those attacks are possible and have been proposed [52], but the lesson learned from this is that even protocols whose theoretical base is proven secure in some scenario are not to be automatically assumed immune to all practical attacks. The attack was made possible because authors have found a way to manipulate random number generator at the receiving station by exploiting weaknesses of single photon detectors. To make things even worse, this strategy made the previously mentioned MIT attack truly viable (not anymore just a simulation).

This is yet another example of importance of (local) RNGs to the security of a cryptographic scheme.

In conclusion, quantum cryptographic protocol BB84 requires that both Alice and Bob posses their private (local) provable random number generators. This is a highly critical requirement. Note that a public server of random numbers can not substitute for local generators because the random numbers would have to be delivered to Alice and Bob in perfect secrecy in the first place, and the server would have to be trusted.

6 Random Numbers in Statistical Cryptography

Statistical cryptography has been invented by U. Maurer in 1991. So called SKAPD protocol [60] resembles quantum cryptography and likewise consists of three subprotocols. In fact the last two subprotocols (the Data reconciliation and the Privacy Amplification) are the same as in QC. However, the first subprotocol, named "Advantage Distillation" (AD) is completely different and it does not involve the quantum channel which potentially makes it much more practical. Instead, it requires something called "binary channel with noise" which is theoretically a classical communication channel complemented with a provable RNG.

The condition for successful key agreement is that prior to AD protocol common information shared by Alice and Bob is greater that common information shared by either Alice and Eve or Bob and Eve.

The practical problem with SKAPD is that it contains an unspoken "zeroth phase" in which Alice and Bob obtain their partially correlated initial strings of bits which satisfy the above condition. There is no known plausible way to make the zeroth phase possible although some scenarios have been proposed (scanning surface of the Moon, listening to noise from far-away galaxies, taking big chunks of internet data, etc.).

7 Random Numbers in Deterministic Cryptography

What we call here "deterministic cryptography" in this Chapter is what is widely known as just "cryptography". Some authors use the name "mathematical cryptography". It is the contemporary cryptography based on the difficulty of computing discrete logarithms in Galois groups and elliptic curve groups, and also the factorization composite number into primes. It also needs and uses random data; an excellent short survey is given in [7]. Since all such security protocols are by definition deterministic and therefore reversible, the only true security resource is that nondeterministic part: a key or one-time

data which is supposed to be "random". Quality and provability of randomness are therefore crucial for security of the whole system.

It is the fact that the deterministic cryptography is the only one in the wider use and that most cryptographers are not aware of or do not care of existence of either quantum cryptography or statistical cryptography because apparently they are not yet practical or sufficiently trusted. Therefore it is important to explore what makes contemporary commercial-grade protocols secure and what could be done to get the maximum security out of them. Our hypothesis is that if a protocol requires random numbers then use of a TRNG maximizes its security. Without ambition to make a strict proof or to give a comprehensive review here let us have a look at several examples supporting this hypothesis.

1. Diffie-Hellman key establishment protocol [19] enables the same functionality as the above mentioned BB84 and SKAPD protocols and is used for example in "https" protocol in order to establish a session key. Protocol requires from both parties (Alice and Bob) to generate private random data, and after some operations send them to each other. More resistant version of DH requires further random data used for digital signatures. A vulnerability of the PRNG built in an early version of Netscape internet browser led to complete compromise of the subsequent cryptographic protocol. An example is attack to the Netscape's 40 bit RC4-40 [75] challenge data and encryption keys, which was able to break https protocol in a minute or so, is described in [28]. The authors of this article stipulate that 128 bit version RC4-128 would not be much harder to break either if seeding is done in a similar fashion.
2. RSA public key protocol relies on generation of public and private keys separately by Alice and Bob. In order to create a private/public pair of keys it is necessary to generate two unique, large prime numbers. Already calculating prime number candidates involves random numbers. After that, candidates need to be tested for primality using the Miller-Rabin algorithm which requires random numbers as the bases in order to properly test for primality. Additional one-time random numbers may be used in the process of actual communication. Where high-entropy physical random bits are not available or are time-expensive (like on a typical PC computer) there is a tendency to "expand" a short random string to a long one by pseudo-random methods. This approach can create serious cryptographic weaknesses because an attacker must guess much smaller number of bits than he would in case of use of truly random numbers.
3. Similarly, a research of cryptographic attack on partially pseudo-random number generator of an AES based commercial cryptographic system is described in [77].

To conclude, in deterministic cryptography random numbers are the only part of the protocol which is different from point to point and furthermore their true randomness is sometimes a prerogative for correct calculations.

Therefore, even though most of deterministic cryptography primitives are not secure, using true random numbers ensures highest achievable security with these methods.

8 Open Problems and Outlook

In this survey article we attempt to show the importance of random numbers for strength of cryptographic protocols not only for quantum and stochastic cryptographies where random numbers are an essential part of data exchanged between communicating parties but also for contemporary deterministic cryptography where unpredictability and maximal entropy of the random numbers used therein maximizes overall cryptographic strength.

True random number generators seem to be in a modest use, even though some companies make a good profit from them [37]. From the available data, it seems that TRNGs are mainly sold to online gambling companies, state security agencies and product labeling and testing industry. At the time of writing of this survey, main problems preventing more widespread use of true random number generators in general are:

1. The lack of generator designs whose proofs of randomness would be at the same time correct, convincing and demonstrated to be resilient to expected imperfections in hardware;
2. The (widespread) lack of understanding that pseudorandomness cannot be used as a substitute for true randomness in so many applications, notably: cryptography both classical and quantum, computer security, Monte Carlo simulations, lottery, testing of products and their functionality and many more.
3. The high price of true random number generators;
4. The lack of support of true random number generators in various popular software that requires random numbers which makes them hard to use.

9 Additional Comments and References

A distinctive difference between PRNG and TRNG is the provability of the latter. Indeed, the only provable feature of a PRNG is that it is not random because all numbers produced thereof can be calculated from a single initial number: the seed. On the other hand TRNGs seem to be inevitably plagued with "small imperfections" in hardware which turn into measurable deviations from randomness which calls for postprocessing. But complex postprocessing blurs or weakens convincingness of eventual randomness proof. Furthermore, a practice of withholding the information on operating principle

of a TRNG as well as a scientific proof of its randomness seem to be almost a rule when it comes to commercial TRNGs. Manufacturers justify this by the need to protect their intellectual property and technology. While such justification is fine when it regards common products (e.g., a dishwasher), it is exactly what ruins the purpose of a TRNG because without a clear insight into the technology and randomness proof of a TRNG one falls back to the nonprovability situation of a PRNG. On the other side of the coin, in scientific publications proofs of randomness are offered very rarely too, probably because the proof is the hard part of the research while it does not seem to be required by editors of scientific journals. Most researchers therefore fall back to the minimum-action strategy: make a TRNG, obtain at least one random number sequence that passes chosen set of randomness tests and publish. However, without a detailed investigation of sensitivity of extracted randomness on small variations in hardware and without randomness proof, a scientific design cannot proceed towards a product. In our view this situation has been improving and will continue to improve very slowly over time, thus ensuring a longevity and freshness of the research of TRNGs.

Even though there is a large collection of publications which document the fact that PRNGs may fail their purpose as random number generators [66, 69, 11, 45, 51, 12, 58, 23, 32, 85, 104, 40, 21, 93] we see that PRNGs are still in much more widespread use than TRNGs even in most critical applications. Among reasons for that is also the fact that PRNGs are so much more convenient, simple and cheap to use than TRNGs but also ubiquitous lack of understanding of what randomness is and what it isn't supported by the non-existence of a widely accepted definition of randomness [45]. Clearly, a further research on that subject is needed.

All commercial TRNGs whose speed is at least 1 Mbit/sec are bulky and the price is in the range $ 5-25k, which is more expensive than most of the software that would use a TRNG. It therefore generally does not pay off to a software manufacturer to make its product much more expensive by requiring a third-party TRNG for generating random numbers. In extreme rare cases though, a software has been married to a selected TRNG: for example Mathematica and Quantis (by a third party) [62].

Commercial TRNGs typically come with drivers that support transfer of random numbers to programming languages such as Pascal or C++ on selected operating systems, e.g., [37], using a product specific subroutine or program library function. This is probably the maximum that a manufacturer can reasonably do to support its product. On the other hand, most of commercial or free software that uses or needs random numbers does not come with support for any TRNG. This means that having a precompiled software there is practically no way to connect it to any TRNG. The only viable solution to include a TRNG in a software package would be to write it from scratch and include in it a specific function calls associated with the specific chosen TRNG. Since there is no industry standard for access to a TRNG from within a computer program (unlike for example to access print-

ers or other common peripherals) one could support only a specific TRNG per programming effort. In our view it is clear that as long as there is no standardized way to access TRNGs, or better yet, until TRNGs are physically integrated in computers and are accessible in major programming languages, their popularity will remain minimal.

While it is clear that true randomness cannot be generated by deterministic operations and that therefore it must rely on physical phenomena, the problem of generating good enough randomness and provability of randomness stay the main open problems with physical RNGs. New directions in development of physical random number generators will probably concentrate on self-calibrating [46, 103] or no-calibration devices [95] with fundamentally random quantum phenomena as a source of randomness.

References

1. V. Bagini and M. Bucci. A design of reliable true random number generator for cryptographic applications. *Cryptographic Hardware and Embedded Systems (CHES)*, Ç. K. Koç and C. Paar, Eds., pages 204-218, Springer 2002.
2. B. Barak, R. Shaltiel, and E. Tromer. True random number generators secure in a changing environment. *Cryptographic Hardware and Embedded Systems (CHES)*, C. D. Walter, Ç. K. Koç, and C. Paar, Eds., pages 166-180, Springer 2003.
3. C. W. J. Beenakker and M. Büttiker. Suppression of shot noise in metallic diffusive conductors. *Phys. Rev. B*, 46:1889-1892, 1992.
4. C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing*, pages 175-179, December 10-12, 1984.
5. C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin. Experimental quantum cryptography. *Journal of Cryptology*, 5(1):3-28, 1992.
6. C. H. Bennett, T. J. Watson, G. Brassard, C. Crepeau, and U. M. Maurer. Generalized privacy amplification. *IEEE Tran. Inform. Theory*, 41(6):1915-1923, November 1995.
7. D. J. Bernstein, J. Buchmann, and E. Dahmen, Eds. *Post-Quantum Cryptography.* Springer 2009.
8. M. Bucci and R. Luzzi. Design of testable random bit generators. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems (CHES)*, J. R. Rao and B. Sunar, Eds., pages 147-156. Springer, 2005.
9. P. Chevalier et al. Random number generator. U.S. Patent Number 3,790,768, February 5, 1974.
10. B. Chor, O. Goldreich, J. Hasted, J. Freidmann, S. Rudich, and R. Smolensky. The bit extraction problem or t-resilient functions. *26th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 396-407, IEEE 1985.
11. T. Click, A. Liu, and G. Kaminski. Quality of random number generators significantly affects results of Monte Carlo simulations for organic and biological systems. *J. Comp. Chem.*, 32:513-524, 2011.
12. P. D. Coddington. Tests of random number generators using Ising model simulations. *Int. J. Mod. Phys. C*, 295303, 1996.
13. Cryptography Research. Evaluation summary: VIA C3 Nehemiah random number generator. `http://www.via.com.tw/en/downloads/whitepapers/initiatives/padlock/evaluation_summary_padlock_rng.pdf`, 2003.

14. Crypt-X. http://roth.cs.kuleuven.be/wiki/Crypt-X.
15. R. B. Davies. Exclusive OR (XOR) and hardware random number generators. http://www.robertnz.net/pdf/xor2.pdf. February 28, 2002.
16. D. Davis, R. Ihaka, and P. P. Fenstermacher. Cryptographic randomness from air turbulence in disk drives. *Advances in Cryptology (Crypto)*, Y. Desmedt, Ed., pages 114-120, Springer, 1994.
17. M. Dichtl. How to predict the output of a hardware random number generator. *Cryptographic Hardware and Embedded Systems (CHES)*, C. D. Walter,Ç. K. Koç, and C. Paar, Eds., pages 181188, Springer 2003.
18. M. Dichtl and J. D. Golic. High-speed true random number generation with logic gates only. *Cryptographic Hardware and Embedded Systems (CHES)*, P. Paillier and I. Verbauwhede, Eds., pages 45-62. Springer 2007.
19. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Tran. Inform. Theory*, 22:644-654, 1976.
20. J. F. Dynes, Z. L. Yuan, A. W. Sharpe, and A. J. Shields. A high speed, postprocessing free, quantum random number generator. *Appl. Phys. Lett.*, 93:031109, 2008.
21. R. J. Easter and C. French. Annex C: Approved Random Number Generators for FIPS PUB 140-2. *Security Requirements for Cryptographic Modules*. NIST, February 2012.
22. ESPACENET. European Patent Office. http://www.espacenet.com.
23. A. M. Ferrenberg, D. P. Landau, and Y. J. Wong. Monte Carlo simulations: hidden errors from 'good' random number generators. *Phys. Rev. Let.*, 69:33823384, 1992.
24. A. Figotin et al. Random number generator based on the spontaneous alpha-decay. U.S. Patent Number 6,745,217, June 1, 2004.
25. V. Fischer and F. Bernard. True random number generators in FPGAs. In Benoit Badrignans, Jean Luc Danger, Viktor Fischer, Guy Gogniat, and Lionel Torres, editors, *Security Trends for FPGAS*, pages 101-135. Springer, 2011.
26. M. Fürst, H. Weier, S. Nauerth, D. G. Marangon, C. Kurtsiefer, and H. Weinfurter. High speed optical quantum random number generation. *Opt. Express*, 18:13029-13037, 2010.
27. I. Gerhardt, Q. Liu, A. Lamas-Linares, J. Skaar, C. Kurtsiefer, and V. Makarov. Perfect eavesdropping on a quantum cryptography system. arXiv:1011.0105v1 [quant-ph], 18 March 2012.
28. I. Goldberg and D. Wagner. Randomness in the Netscape browser. *Dr. Dobb's Journal*, January 1996.
29. L. Gollub. Vorrichtung zur gewinnung von zufallszahlen. Germany Patent Number DE19743856A1, April 8, 1999.
30. M. Goresky and A. Klapper. *Algebraic Shift Register Sequences*. Cambridge University Press, 2012.
31. D. Gottesman, H.-K. Lo, N. Lutkenhaus, and J. Preskill. Security of quantum key distribution with imperfect devices. *Quantum Information and Computation*, 4:325-360, 2004.
32. P. Grassberger. On correlations in good random number generators. *Phys. Lett. A*, 181:43-46, 1993.
33. H. Guo, W. Tang, Y. Liu, and W. Wei. Truly random number generation based on measurement of phase noise of a laser. *Phys. Rev. E*, 81:051137, 2010.
34. L. Hars. Electronic circuit for random number generation. US Patent US7315874(B2), 2008.
35. R. Heinen. Private communication. University of Twente, Twente, Netherlands
36. P. Hellekalek. Good random number generators are (not so) easy to find. *Mathematics and Computers in Simulation*, 46:485-505, 1998.
37. IdQuantique. Quantis: True random number generator exploiting quantum physics. http://www.idquantique.com/random-number-generators/products/products-overview.html, 2012.

38. Institut Ruder Bošković. QRBG 121. http://qrbg.irb.hr, 2012.
39. T. Jennewein, U. Achleitner, G. Weihs, H. Weinfurter, and A. Zeilinger. A fast and compact quantum random number generator. *Rev. Sci. Instrum.*, 71:1675-1680, 2000.
40. P. Jonsson. Boom in Internet gambling ahead? US policy reversal clears the way. http://tinyurl.com/86b9aaz, December 26, 2011.
41. B. Jun and P. Kocher. The Intel random number generator. Cryptography Research Inc., White Paper Prepared for Intel Corporation, April 22, 1999.
42. I. Kanter, Y. Aviad, I. Reidler, E. Cohen, and M. Rosenbluh. An optical ultrafast random bit generator. *Nature Photonics*, 4(1):58-61, December 2010.
43. J. Kelsey, B. Schneier, D. Wagner, and C. Hall. Cyptanalytic attacks on pseudorandom number generators. *Fast Software Encryption*, pages 168-188, Springer 2005. pages 53-63, Springer, 2005.
44. T. Kim, I. S. Wersborg, F. N. C. Wong, and J. H. Shapiro. Complete physical simulation of the entangling-probe attack on the Bennett-Brassard 1984 protocol. *Phys. Rev. A*, 75:042327, 2007.
45. D. E. Knuth. High speed single photon detection in the near infrared. *The Art of Computer Programming*, Vol. 2, 3rd Edition, Addison Wesley, 1997.
46. O. Kwon, YQuantum random number generator using photon-number path entanglement. *Appl. Optics*, 48:1774-1778, 2009.
47. P. Li, Y. C. Wang, and J. Z. Zhang. All-optical fast random number generator. *Opt. Express*, 18:20360-20369, 2010.
48. P. Lacharme. Post processing functions for a biased physical random number generator. *Fast Software Encryption (FSE)*, pages 334-342, 2008.
49. P. Lacharme. Analysis and Construction of Correctors. *IEEE Trans. Information Theory*, 55(10):4742-4748, October 2009.
50. X. Li, A.B. Cohen, T.E. Murphy, and R. Roy. Scalable parallel physical random number generator based on a superluminescent LED. *Opt. Lett.*, 36:1020-1022, 2011.
51. Lotteries and Gaming Authority. Remote gaming regulations, Legal notice 176 of 2004, 110 of 2006, 2760 and 426 of 2007, and 90 of 2011. Malta, 2011.
52. L. Lydersen, V. Makarov, and J. Skaar. Secure gated detection scheme for quantum cryptography. arXiv:1101.5698 [quant-ph], 29 January 2011.
53. L. Lydersen, C. Wiechers, C. Wittmann, D. Elser, J. Skaar, and V. Makarov. Hacking commercial quantum cryptography systems by tailored bright illumination. *Nature Photonics* , 4:686689, 2010.
54. G. Marsaglia. DIEHARD battery of stringent randomness tests. http://stat.fsu.edu/~geo/diehard.html.
55. G. Marsaglia and W. W. Tsang. The ziggurat method for generating random variables. *Journal of Statistical Software*, 5(8):1-7, October 2000. http://www.jstatsoft.org/v05/i08.
56. J. L. Massey. Shift-register synthesis and BCH decoding. *IEEE Tran. Inform. Theory*, 15(1):122-127, January 1969.
57. M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8:3-30, 1998. http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html.
58. A. De Matteis and S. Pagnutti. Long-range correlations in linear and non-linear random number generators. *Parallel Comput.*, 14(2):207210, June 1990.
59. U. M. Maurer. A universal statistical test for random bit generators. *Journal of Cryptology*, 5(2):89-105, 1992.
60. U. Maurer. Secret key agreement by public discussion from common information. *IEEE Tran. Inform. Theory*, 39:733-742, 1993.
61. T. McNichol. Totally random. *Wired*, 11(8), August 2003. http://www.wired.com/wired/archive/11.08/random.html.
62. J. A. Miszczak. Generating and using truly random quantum states in Mathematica. arXiv:1102.4598v2 [quant-ph], 19 October 2011.

63. J. von Neumann. Various techniques for use in connection with random digits. *John von Neumann Collected Works*, Vol. 5, pages 768770, 1963.
64. H. Nyquist. Thermal agitation of electric charge in conductors. *Phys. Rev.*, 32:110-113, 1928.
65. oelermans R. Oelermans and V. Miche Digital true random number generator circuit. US Patent Application, US2002156819 (A1), 2002.
66. G. Parisi and F. Rapuano. Effects of the random number generator on computer simulations. *Physics Letters B*, 157:301-302, 1985.
67. Y. Peres. Iterating von Neumann's procedure for extracting random bits. *Ann. Stat.* , 20:590-597, 1992.
68. PicoQuant. PQRNG 150. `http://www.picoquant.com/products/pqrng150/pqrng150.htm`, 2012.
69. A. Proykova. How to improve a random number generator. *Comp. Phys. Comm.*, 124:125-131, 2000.
70. B. Qi, Y.-M. Chi, H.-K. Lo, and L. Qian. High speed quantum random number generation by measuring phase noise of asingle mode laser. *Opt. Letters*, 35:312-314, 2010.
71. qutools GmbH. quRNG. `http://www.qutools.com/products/quRNG/`, 2012.
72. J. A. Reeds and N. J. A. Sloane. Shift-register synthesis (Modulo m). *SIAM J. Comput*, 14:505-513, 1985.
73. I. Reidler, Y. Aviad, M. Rosenbluh, and I. Kanter. Ultra high-speed random number generation based on a chaotic semiconductor laser. *Phys. Rev. Lett.*, 103(2):024102, 2009.
74. T. Ritter. Random number machines: A literature survey. `http://www.ciphersbyritter.com/RES/RNGMACH.HTM`, December 4, 2002.
75. R. L. Rivest. The RC4 Encryption Algorithm. RSA Data Security Inc., March 1992.
76. F. Rodriguez-Henriquez, N. A. Saqib, A. Diaz-Perez, and Ç. K. Koç. *Cryptographic Algorithms on Reconfigurable Hardware*. Springer, 2007
77. C. B. Roellgen. Visualisation of potential weakness of existing cipher engine implementations in commercial on-the-fly disk encryption software. Global IP Telecommunications, Ltd. & PMC Ciphers, Inc., August 15, 2008.
78. A. Ruhkin. Statistical testing of randomness: Old and new procedures. *Randomness through Computation*, H Zenil, Ed., World Scientific, 2011.
79. A. Ruhkin et al. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800-22rev1a, April 2010.
80. D. Schellekens, B. Preneel, and I. Verbauwhede. FPGA Vendor agnostic true random number generator. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.86.5319`, 2006.
81. W. Schindler. Evaluation criteria for physical random number generators. In Ç. K. Koç, editor, *Cryptographic Engineering*, pages 25-54. Springer, 2009.
82. W. Schindler. Random number generators for cryptographic applications. In Ç. K. Koç, editor, *Cryptographic Engineering*, pages 5-23. Springer, 2009.
83. W. Schindler. Anwendungshinweise und Interpretationen zum Schema (AIS). AIS 32, Version 1, Bundesamt für Sicherheit in der Informationstechnik, 2001.
84. W. Schindler and W. Killmann. Evaluation criteria for true (physical) random number generators used in cryptographic applications. *Cryptographic Hardware and Embedded Systems (CHES)*, B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, Eds., pages 431449, Springer 2002.
85. F. Schmid and N. B. Wilding. Errors in Monte Carlo simulations using shift register random number generators. *Int. J. Mod. Phys.*, 6:781787, 1995.
86. R. Shaltiel. Recent developments in explicit constructions of extractors. *Bull. EATCS*, 77:6795, 2002.
87. R. Shaltiel. How to get more mileage from randomness extractors. *Random Struct. Algorithms*, 33:157-186, 2008.

88. P. Shor and J. Preskill. Simple proof of security of the BB84 quantum key distribution protocol. *Phys. Rev. Lett.*, 85:441-444, 2000.
89. A. Sidorenko and B. Schoenmakers. State recovery attacks on pseudorandom generators. *Western European Workshop on Research in Cryptology*, pages 53-63, Springer, 2005.
90. A. Stefanov, N. Gisin, O. Guinnard, L. Guinnard, and H. Zbinden. Optical quantum random number generator. *J. Mod. Opt.*, 47, 595-598, 2000.
91. M. Stipčević. Apparatus and method for generating true random bits based on time integration of an electronic noise source. WIPO Patent Number WO03040854, October 17, 2001.
92. M. Stipčević. Fast nondeterministic random bit generator based on weakly correlated physical events. *Rev. Sci. Instrum.*, 75:4442-4449, 2004.
93. M. Stipčević. Quantum random bit generator. WIPO Patent Number WO2005106645 (A2), April 30, 2004.
94. M. Stipčević. Preventing detector blinding attack and other random number generator attacks on quantum cryptography by use of an explicit random number generator. Manuscript in preparation, 2012.
95. M. Stipčević and B. M. Rogina. Quantum random number generator based on photonic emission in semiconductors. *Rev. Sci. Instrum.*, 78:1-7, 2007.
96. B. Sunar. True random number generators for cryptography. In Ç. K. Koç, editor, *Cryptographic Engineering*, pages 55-73. Springer, 2009.
97. B. Sunar and D. Schellekens. Random number generators for integrated circuits and FPGAs. In I. Verbauwhede, editor, *Secure Integrated Circuits and Systems*, pages 107-124. Springer, 2010.
98. B. Sunar, W. J. Martin, and D. R. Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Trans. on Computers*, 56(1):109-119, January 2007.
99. S. Takagi. Random number data generator. US Patent US2003208517, 2003.
100. G. Taylor and G. Cox. Behind Intel's new random-number generator. *IEEE Spectrum*, `http://spectrum.ieee.org/computing/hardware/behind-intels-new-randomnumber-generator`, August 24, 2011.
101. T. E. Tkacik. A hardware random number generator. *Cryptographic Hardware and Embedded Systems (CHES)*, B. S. Kaliski Jr., Ç. K. Koç, and C. Paar, Eds., pages 450453, Springer 2002.
102. A. Uchida et. al. Fast physical random bit generation with chaotic semiconductor lasers. *Nature Photon.*, 2:728-732, 2008.
103. G. Vallone, D. Marangon, M. Tomasin, and P. Villoresi Self-calibrating quantum random number generator based on the uncertainty principle. arXiv:1401.7917 [quant-ph], 30 January 2014.
104. I. Vattulainen, T. Ala-Nissila, and K. Kankaala. Physical tests for random numbers in simulations. *Phys. Rev. Lett.*, 73:25132516, 1994.
105. VIA Inc. Via security application note. `www.via.com.tw/en/downloads/whitepapers/initiatives/padlock/security_application_note.pdf`, 2005.
106. VIA Inc. AES encryption. `http://www.via.com.tw/en/initiatives/padlock/hardware.jsp`, 2012.
107. VIA Inc. Random number generation. `http://www.via.com.tw/en/initiatives/padlock/hardware.jsp`, 2012.
108. VIA Inc. Via padlock security engine. `http://www.via.com.tw/en/initiatives/padlock/hardware.jsp`, 2012.
109. J. Viega. Practical random number generation in software. In *Proceedings of 19t Annual Computer Security Applications Conference*, pages 129-140, 2003.
110. C. H. Vincent. The generation of truly random binary numbers. *J. Phys. E: Scientific Instruments*, 3:594-598, 1970.

111. M. Wahl, M. Leifgen, M. Berlin, T. Roehlicke, H.J. Rahn, and O. Benson. An ultrafast quantum random number generator with provably bounded output bias based on photon arrival time measurements. *Appl. Phys. Lett.*, 98:171105, 2011.
112. J. Walker. Ent: A pseudorandom number sequence test program. `http://www.fourmilab.ch/random/`.
113. C. S. Wallace. Physically random generator. *Jour. Comp. Sys. Sci. and Eng.*, 5(2):82-88, 1990.
114. A. B. Wang, Y. C. Wang, and H. C. He. Enhancing the bandwidth of the optical chaotic signal generated by a semiconductor laser with optical feedback. *IEEE Photon. Tech. Lett.*, 20:1633-1635, 2008.
115. A. B. Wang, Y. C. Wang, and J. F. Wang. Route to broadband chaos in a chaotic laser diode subject to optical injection. *Opt. Lett.*, 34:1144-1146, 2009.
116. M. A. Wayne, E. R. Jeffrey, G. M. Akselrod, and P. G. Kwiat. Photon arrival time quantum random number generation. *J. Mod. Opt.*, 56:516522, 2009.
117. M. A. Wayne and P. G. Kwiat. Low-bias high-speed quantum random number generator via shaped optical pulses. *Opt. Express*, 18:9351-9357, 2010.
118. S.-K. Yoo, D. Karakoyunlu, B. Birand, and B. Sunar. Improving the robustness of ring oscillator TRNGs. *ACM Transactions on Reconfigurable Technology and Systems*, 3(2):9, May 2010.
119. Z. L. Yuan, B. E. Kardynal, A. W. Sharpe, and A. J. Shields. High speed single photon detection in the near infrared. *Appl. Phys. Lett.*, 91:041114, 2007.