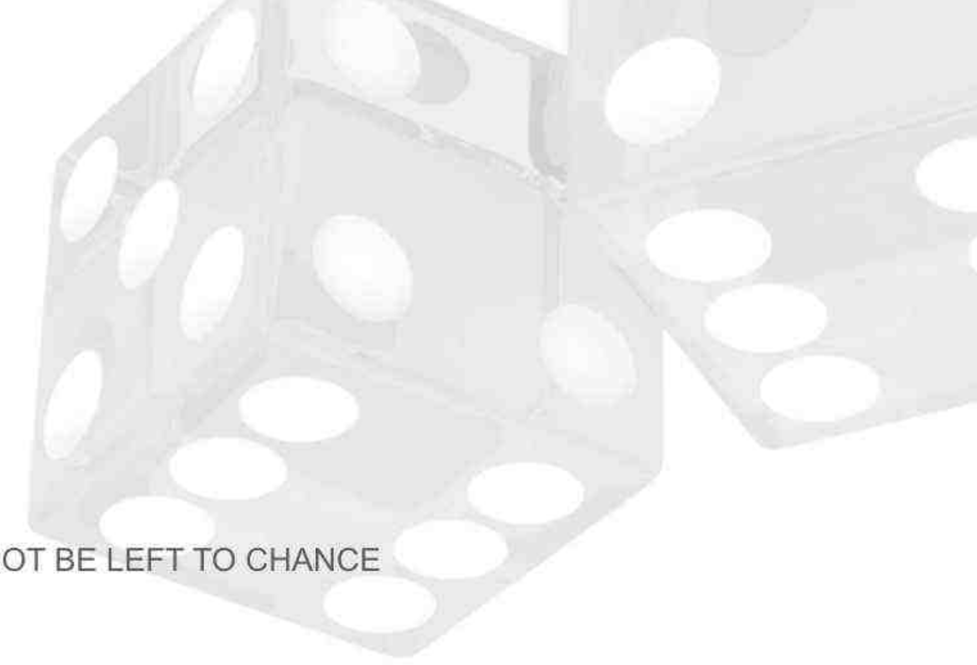


FROM VISION TO TECHNOLOGY

REDEFINING RANDOMNESS

QUANTIS

WHEN RANDOM NUMBERS CANNOT BE LEFT TO CHANCE

ID Quantique White Paper

RANDOM NUMBER GENERATION USING QUANTUM PHYSICS

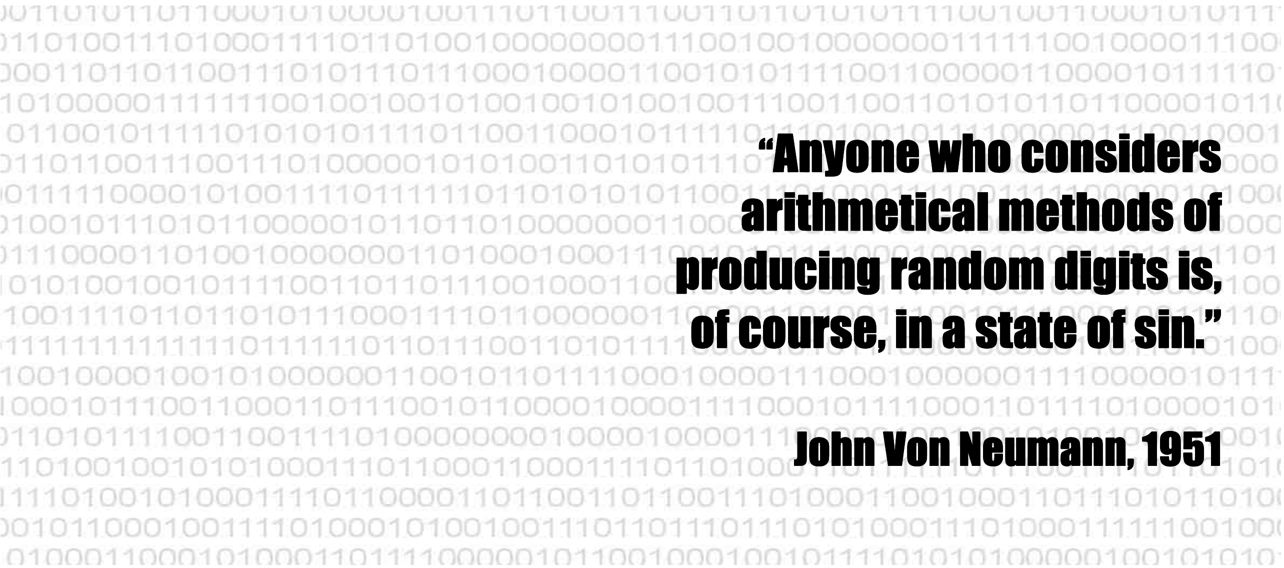
Version 3.0

April 2010

Table of contents

ID Quantique SA	Tel:	+41 (0)22 301 83 71
Ch. de la Marbrerie, 3	Fax:	+41 (0)22 301 83 79
1227 Carouge		www.idquantique.com
Switzerland		info@idquantique.com

Information in this document is subject to change without notice.



> **“Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin.”**
>
> **John Von Neumann, 1951**

Summary

The generation of truly random numbers plays a critical role in a number of important applications. In cryptography, in the field of numerical simulations and in the gaming industry, just to name a few, high-quality random numbers are absolutely vital. Unfortunately, their generation is often overlooked.

This white paper first gives an overview of the applications requiring random numbers and discusses the definition of randomness. Random number generators are then presented, starting with software generators and later considering physical generators. Processes based on classical and quantum physics are compared, to establish the superiority of quantum random number generators. Finally, a new random number generator based on quantum physics, called Quantis, is presented and its features discussed.

Applications of random numbers

Random numbers are useful in countless applications, which have evolved over time. With the expansion of computers fields of use and the rapid development of electronic communication networks in the past fifty years, the number of such applications is growing quickly. The following subsections non-exhaustively list examples of such applications.

Cryptography

Cryptography can be defined as "the art and science of keeping messages secure" [1]. It consists of algorithms and protocols that can be used to ensure the confidentiality, the authenticity and the integrity of communications. Cryptographic algorithms come in a variety of flavors. Some are difficult to crack but make substantial demands to processing power and key management. Others are easier to crack but less demanding and therefore better suited for some applications. All strong cryptography requires true random numbers to generate keys, but how many depends on the encryption scheme. The strongest possible method, One Time Pad (OTP for short) encryption, is the most demanding of all; it requires as many random bits as there are bits of information to be encrypted. Many security applications have failed or been severely compromised because their random number generators failed to be sufficiently random.

Confidentiality

In order to guarantee the confidentiality of a message, the sender combines the plain text with a key using an encryption algorithm to obtain the cipher text. This cipher text is then sent over an insecure communication channel to the recipient, who uses a decryption algorithm and a key to unscramble it and recover the plain text. In an ideal cipher system, it is impossible for an eavesdropper to decrypt the cipher text without the key.

The strength of a cipher system ultimately depends on the strength of the key used or equivalently on the difficulty for an eavesdropper to guess it. This difficulty clearly increases with the key length – typical key sizes currently in use are 56 bits (DES), 168 bits (3-DES) and 256 bits (IDEA or AES) – and its unpredictability, which is a function of the randomness of the number used to generate the key. Consequently, it is essential to use sufficiently long and truly random numbers for key generation.

Authentication

Authentication is essential, when a client logs on to a server, to ensure that he is actually authorized to access the data. In order to illustrate the use of random numbers in authentication, one example of a protocol is schematically described.

The simplest possibility for a client to authenticate itself would be to send his password to the server. Doing this is however dangerous, because the password travels over the network and can be intercepted. In order to avoid this, the server, which keeps a copy of all the passwords, sends a message containing a random number to the client. The client calculates a function of this message and his password, and sends the result back to the server. The server compares this result with the value it computed itself using a copy of the password.

The fact that the message sent by the server to the client contains a random number that is never reused, prevents a so-called replay attack, where an adversary would record the authentication messages and resend them to the server, every time it wants to log on. Note that in this scheme, the password is never sent across the network.

Although authentication schemes are in practice more complex, they often use random numbers.

Scientific calculations

Scientists have devised techniques relying on random numbers to model and simulate complex systems. These techniques are fast and yield high accuracy results. They are essential for modern numerical simulations.

Lotteries and gambling

In games of chance, it must not be possible for a player to increase his probability to win by discovering a bias towards certain outcomes in the game procedure. Modern lotteries and gambling machines are all based on the use of random numbers to guarantee a uniform winning probability.

What is a random number?

Although it may look simple at first sight to give a definition of what a random number is, it proves to be quite difficult in practice.

A random number is a number generated by a process, whose outcome is unpredictable, and which cannot be subsequentially reliably reproduced. This definition works fine provided that one has some kind of a black box – such a black box is usually called a random number generator – that fulfills this task.

However, if one were to be given a number, it is simply impossible to verify whether it was produced by a random number generator or not. In order to study the randomness of the output of such a generator, it is hence absolutely essential to consider sequences of numbers.

It is quite straightforward to define whether a sequence of infinite length is random or not. This sequence is random if the quantity of information it contains – in the sense of Shannon's information theory – is also infinite. In other words, it must not be possible for a computer program, whose length is finite, to produce this sequence. Interestingly, an infinite random sequence contains all possible finite sequences. Such an infinite sequence does for example contain the Microsoft Windows source code or the text of the Geneva conventions. Unfortunately, this definition is not very useful, as it is not possible in practice to produce and process infinite sequences.

In the case of a finite sequence of numbers, it is formally impossible to verify whether it is random or not. It is only possible to check that it shares the statistical properties of a random sequence – like the equiprobability of all numbers – but this a difficult and tricky task. To illustrate this, let us for example consider a binary random number generator producing sequences of ten bits. Although it is exactly as likely as any other ten bits sequences, 1 1 1 1 1 1 1 1 1 1 does look less random than 0 1 1 0 1 0 1 0 0 0.

In order to cope with this difficulty, definitions have been proposed to characterize "practical" random number sequences. According to Knuth [2], a sequence of random numbers is a sequence of independent numbers with a specified distribution and a specified probability of falling in any given range of values. For Schneier [1], it is a sequence that has the same statistical properties as random bits, is unpredictable and cannot be reliably reproduced. A concept that is present in both of these definition and that must be emphasized is the fact that numbers in a random sequence must not be correlated. Knowing one of the numbers of a sequence must not help predicting the other ones. Whenever random numbers are mentioned in the rest of this paper, it will be assumed that they fulfill these "practical" definitions.

Testing randomness

Statistical randomness tests aim at determining whether a particular sequence of numbers was produced by a random number generator. The approach is to calculate certain statistical quantities and compare them with average values that would be obtained in the case of a random sequence. These average values are obtained from calculations performed on the model of an ideal random number generator. Testing randomness is an empirical task. There exists numerous tests, each one of them revealing a particular type of imperfection in a sequence.

One example is the frequency test. In the case of binary sequences, it focuses on the relative frequency of 1's with respect to 0's. The autocorrelation test, which investigates correlations between adjacent bits, is another example. A good reference on randomness testing can be found in [2]. Maurer demonstrated that all tests can be derived from trying to compress a sequence [3]. If a given sequence can be compressed, then it is not random. When put into the perspective of the definition given above for an infinite random sequence, this observation is natural.

Because of the difficulty of defining what a random number is, it is essential to choose an adequate generator to produce these numbers. Moreover, it is safer to have a good understanding of the underlying randomness generating process. This white paper further discusses different types of random number generators – both software and hardware – and presents in more detail the Quantis generator, which is based on a quantum process.

Generating random numbers

A random number generator is a device that produces sequences of numbers complying with the definitions proposed above. There exist two main classes of generators: software and physical generators. From a general point of view, software generators produce so-called pseudo random numbers. Although they may be useful in some applications, they should not be used in most applications where randomness is required. These two classes, as well as their respective advantages, are discussed below.

Software solutions

Computers are deterministic systems. Given a certain input, a program will always produce the same output. Because of this very fundamental property, it is impossible for a program to produce a sequence of random numbers. The sequence may have some of the properties of a random sequence, and thus pass some statistical randomness tests, but it is always possible to reproduce it. As the sequences they produce look like random sequences, these generators are called pseudo-random number generators. It is however clear that they do not fulfill the definitions given at the beginning of this white paper.

Pseudo-random number generators consist of an algorithm into which some initial value – it is called the seed – is fed and which produces by iteration a sequence of pseudo-random numbers. More information on some pseudo-random number generators can be found in [2]. Although their period can be made very long, the sequence produced by such a generator is always periodic. When working with large sets of random numbers or long sequences, it is important to verify that the period is large enough. One of the properties of the sequences produced in this way is that, as soon as one element of the sequence is know, all the other elements of the sequence, both preceding and following, can be determined. It is immediately obvious that this property is especially critical when such numbers are used in cryptography for key generation. The sequences produced by a good pseudo-random generator initialized with an appropriate seed, pass most statistical tests. However, it is also important to realize that if the sequence considered is long enough, it will always fail at least some tests, because of periodicity.

One important issue when using pseudo-random number generators is the choice of the seed value. If one does not want to continuously cycle through the same sequence, it is essential to change periodically the seed value. How should this seed value be chosen? Ideally, it should be random. As a pseudo-random generator is used, it is likely that no random number generator is available. It is a catch-22 situation. A solution, which is not always satisfactory, is to use entropy gathering. System information – the clock, the time interval between keystrokes, etc - is combined to produce a seed. This technique however requires extreme caution.

A security problem encountered by Netscape with its browser in 1995 illustrates the risks associated with the use of pseudo-random numbers in cryptography [4]. At the time, the web was full of promises for online commerce and Netscape had developed a protocol, called SSL and still in use today, to secure communications over the web. Like in the case of other cryptographic protocols, the security of SSL crucially depends on the unpredictability of the key. The company implemented this protocol in its browser, but relied on a pseudo-random number generator for key generation. Two Berkeley graduate students reverse-engineered the code of the browser and revealed a serious security flaw. They noticed that the seed used by the pseudo-random number generator depended on the time of the day and some system information (the process ID and the parent process ID). They showed that it was relatively easy to guess these quantities, and thus to reduce the number of possible keys that one should try to crack the protocol. This attack reduced the time for a brute force attack of the protocol from more than thirty hours to a few minutes, and as little as a few seconds in some special cases. This security flaw illustrates why pseudo-random number generators are usually considered inapropriate for high-security cryptographic applications.

In spite of the fact that they do not produce random numbers, these generators do have some advantages. First, their cost is virtually zero, as they can be implemented in software and numerous libraries are freely available. Second, their main disadvantage – namely that the sequences they produce are reproducible – can in certain cases represent an advantage. When using random numbers for scientific calculations, it is sometimes useful to be able to replay a sequence of numbers when debugging the simulation program. This is one of the reasons – the other being the lack of good physical random numbers generators – why these generators are widely used in these applications. However, it is important, even in this field, to remain careful when using pseudo-random numbers. They have been shown to cause artifacts in certain simulations. A good approach is to use pseudo-random numbers when debugging, and then resort to a physical generator to refine and confirm the simulation.

Physical sources of randomness

In applications where pseudo-random numbers are not appropriate, one must resort to using a physical random number generator. When using such a generator, it is essential to consider the physical process used as the randomness source. This source can be either based on a process described by classical physics or by quantum physics. Classical physics is the set of theories developed by physicists before the beginning of the XXth century and which describes macroscopic systems like falling coins. Quantum physics is a set of theories elaborated by physicists during the first half of the XXth century and which describes microscopic systems like atoms or elementary particles. Some examples of generators based on each of these theories, along with their advantages, are presented below, after a brief discussion of biased random number sequences.

Biased and unbiased sequences

A problem encountered with physical random number generators is their bias. A binary generator is said to be biased when the probability of one outcome is not equal to the probability of the other outcome. Bias arises because of the difficulty to devise precisely balanced physical processes. It is however less of a problem than one might expect at first sight. There exists some post-processing algorithm that can be used to remove bias from a sequence or random numbers.

The simplest of these unbiasing procedures was first proposed by Von Neumann [5]. The random bits of a sequence are grouped in subsequences of two bits. Whenever the two bits of a subsequence are equal, it is discarded. When the two bits are different and the subsequence starts with a 1, the subsequence is replaced by a 1. When it starts with a 0, it is replaced by a 0. After this procedure, the bias is removed from the sequence.

The cost of applying an unbiasing procedure to a sequence is that it is shortened. In the case of the Von Neumann procedure, the length of the unbiased sequence will be at most 25% of the length of the raw sequence. It was mentioned above that randomness tests basically all amount to verifying whether the sequence can be compressed. An unbiasing procedure can be seen as a compression procedure. After its application, the bias is removed and no further compression is possible, guaranteeing that the sequence will pass the tests. Other unbiasing procedures exist. The one proposed by Peres [6] for example is significantly more efficient than the Von Neumann procedure.

Processes described by classical physics – determinism hidden behind complexity

Macroscopic processes described by classical physics can be used to generate random numbers. The most famous random number generator – coin tossing – indeed belongs to this class. However, it is very important to realize that classical physics is fundamentally deterministic. The evolution of a system described by classical physics can be predicted, assuming that the initial conditions are known. In the case of a coin, a physicist knowing precisely its weight, its initial position, the force applied to it by the hand, the speed of the wind, as well as all the other relevant parameters, should in principle be able to predict the outcome of the throw. Why is that, that in practice this prediction is not possible? Coin tossing is a chaotic process. Chaos is a type of behavior observed in systems whose evolution exhibits extreme sensitivity to initial conditions. Coin tossing is not the only physical system with chaotic evolution. Turbulences in a flow (turbulences in a lava lamp have been used to generate random numbers [7]) or meteorological phenomena are good examples of chaotic systems. The evolution of these systems is so sensitive to initial conditions that it is simply not possible to determine them precisely enough to allow reliable prediction of future transformations.

In spite of its popularity, coin tossing is clearly not very practical when many random events are required. Other examples of physical random number generators based on chaotic processes include the monitoring of an electric noise current in a resistor or in a Zener diode. Formally the evolution of these generators is not random, but just very complex. One could say that determinism is hidden behind complexity.

Although their random numbers are likely to pass randomness tests, these generators are difficult to model. This means that it is impossible to verify, while acquiring numbers, that they are operating properly. In addition, it is difficult to ensure that the system is not interacting – even in a subtle way – with its environment, which could alter the quality of its output.

Processes described by quantum physics – randomness revealed by simplicity

Contrary to classical physics, quantum physics is fundamentally random. It is the only theory within the fabric of modern physics that integrates randomness. This fact was very disturbing to physicists like Einstein who invented quantum physics. However, its intrinsic randomness has been confirmed over and over again by theoretical and experimental research conducted since the first decades of the XXth century.

When designing a random number generator, it is thus a natural choice to take advantage of this intrinsic randomness and to resort to the use of a quantum process as source of randomness. Formally, quantum random number generators are the only true random number generators. Although this observation may be important in certain cases, quantum random number generators have other advantages. This intrinsic randomness of quantum physics allows selecting a very simple process as source of randomness. This implies that such a generator is easy to model and its functioning can be monitored in order to confirm that it operating properly and is actually producing random numbers. Contrary to the case where classical physics is used as the source of randomness and where determinism is hidden behind complexity, one can say that with quantum physics randomness is revealed by simplicity.

Until recently the only quantum random number generator that existed were based on the observation of the radioactive decay of some element. Although they produce numbers of excellent quality, these generators are quite bulky and the use of radioactive materials may cause health concerns. The fact that a simple and low cost quantum random number generators did not exist prevented quantum physics to become the dominant source of randomness.

The *Quantis* Quantum Random Number Generator

In 2001, ID Quantique introduced the first commercial quantum random number generator, which generated a strong interest. Quantis is a second generation quantum random number generator exploiting an optical quantum process as source of randomness. Quantis is unique in that it relies on quantum physics to produce a high bit rate of 4 to 16 Mbits/sec of truly random bits. The product comes as a USB device, a PCI Express (PCIe) card, PCI card as well as an OEM component. It is possible to download random numbers produced by a Quantis quantum random number generator by visiting www.randomnumbers.info. This section describes the functioning and the features of Quantis.

Principle

Optics is the science of light. From a quantum physics point of view, light consists of elementary "particles" called photons. Photons exhibit in certain situations a random behavior. One such situation, which is very well suited to the generation of binary random numbers, is the transmission upon a semi-transparent mirror. The fact that a photon incident on such a component be reflected or transmitted is intrinsically random and cannot be influenced by any external parameters. Figure 1 schematically shows this optical system.

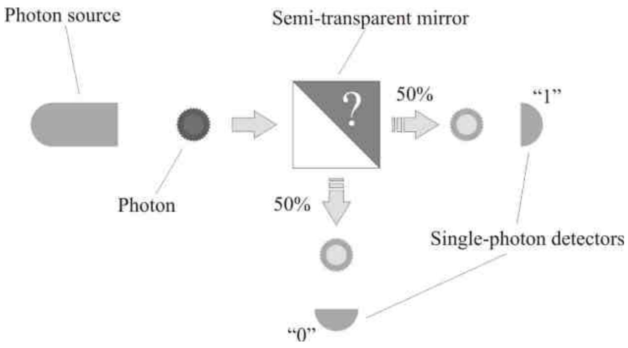


Figure 1: Optical system used to generate random numbers.

Figure 2 shows the block diagram of the Quantis random number generator. It consists of three subsystems. The first one is the core of the generator and contains the optical elements that are used to implement the random process and produce the random outcomes. It comprises a light emitting diode producing the photons, a transmission element, where the random process takes place, and two single-photon detectors – detectors with single-photon resolution – to record the outcomes. The optical subsystem is controlled by a synchronization and acquisition electronic circuit. This subsystem comprises a clock and triggering electronics for the photon source, as well the acquisition electronics for the single-photon detectors. The processing and interfacing subsystem perform statistical and hardware checks, as well as unbiasing of the sequence. These operations are discussed in more details in the following subsection. This subsystem also shapes the output electronic signals.

Unbiasing of the random numbers

As mentioned above, physical processes are difficult to precisely balance. It is thus difficult to guarantee that the probability of recording a 0, respectively a 1, are exactly equal to 50%. With Quantis, the difference between these two probabilities is smaller than 10% - or equivalently the probabilities are comprised between 45% and 55%. As this bias may not be acceptable in certain applications, the processing unit of Quantis performs unbiasing of the sequence.

Status Monitoring

As discussed above, one of the main advantages of quantum random number generators is that they are based on a simple and fundamentally random process that is easy to model and monitor. The processing unit of Quantis performs a live verification of its functioning. It continuously checks that the light source and the two detectors are correctly working, and that the raw output stream statistics are within certain bounds. A status bit is output by Quantis. If all the conditions are fulfilled, this bit is equal to 1. If one of the conditions is not fulfilled, the status bit is set to 0 and the bit stream is inhibited. Thanks to this feature, the users of Quantis can have a high level of trust in the random numbers they are using.

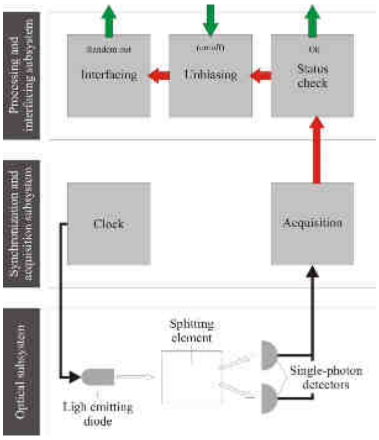


Figure 2: Block diagram of the Quantis RNG.

Packaging

Quantis is available as a component, in the form of a compact metal package that can be mounted on plastic circuit boards (see Figure 3). It is also available as a USB device, a PCI Express (PCIe) card and a PCI card, that can be installed in a computer. Refer to www.idquantique.com for more information on Quantis.

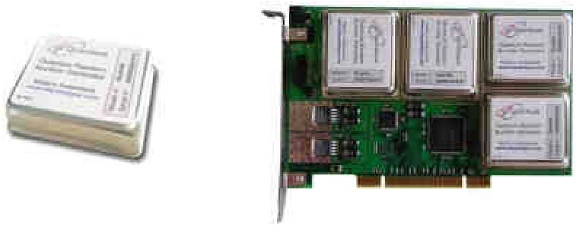


Figure 3: The Quantis random number generator, available for example as an OEM component (left) and as a PCI-card (right), offers high-quality random numbers at a speed of up to 16 Mbits/sec.

Conclusion

Random number generation is a critical security and reliability criterion in many demanding applications. Because of its intrinsic randomness, quantum physics is an excellent source of randomness. Quantis is a compact, low cost and easy to use random number generator exploiting a quantum optical process as source of randomness. It features a high bit rate output stream – up to 16 Mbits/s – which does not exhibit any correlations and passes all statistical tests. When the generation of random numbers cannot be left to chance, the use of Quantis is the solution.

References

[1] Schneier, B., *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley & Sons, New York, (1996).

[2] Knuth, D., *The Art of Computer Programming*, Vol. 2, 2nd ed., Addison-Wesley, Reading, (1981).

[3] Maurer, U., "A universal statistical test for random bits generators", *Journal of Cryptology*, 5, 89-106 (1992).

[4] Markoff, J., "Security flaw is discovered in software used in shopping", *The New York Times* (19 September 1995)

[5] Von Neumann, J., "Various techniques used in connection with random digits", *Applied Mathematics Serires*, no. 12, 36-38 (1951).

[6] Peres, Y., *Ann. Stat.*, 20, 590 (1992).

[7] www.lavarand.org