Journal of Applied Mathematics & Bioinformatics, vol.5, no.3, 2015, 155-176
ISSN: 1792-6602 (print), 1792-6939 (online)
Scienpress Ltd, 2015

Design and evaluation of random number generators

George Marinakis[1]

Abstract

In a cryptographic system the most secret component is the key. Therefore, a great caution must be given in the key management, which concerns how the keys are produced, loaded, renovated and distributed. Cryptographic keys are generated using various types of random number generators (RNG). If these RNG are not secure, they will constitute the weakest point of the cryptosystem, which might be susceptible to various attacks. In this study we examine the basic components and the security weaknesses of deterministic and non-deterministic RNGs and we propose some improvements in their design methods and evaluation procedures.

Mathematics Subject Classification: 65C10; 94A60; 68P25

Keywords: Random Number Generation; Cryptography; Data encryption

[1] Telecommunications and Electronics School of Military Signal Officers, Athens, Greece. E-mail: gmari@tee.gr

Article Info: *Received* : November 14, 2014. *Revised* : February 15, 2015.
Published online : September 15, 2015.

1 Random Number Generators (RNG)

In a random number sequence all the possible numbers appear in a random order and there is no way of predicting the occurrence of one number from its preceding or its following numbers. Random Number Generator (RNG) is a hardware or software device which generates number sequences that are indistinguishable from truly random numbers.

There is a great need for random numbers generation in a big number of applications, such as gambling, statistical sampling, computer simulations, Monte Carlo simulations, stochastic simulations, authentication protocols, training of neural networks, random padding, passwords, initialization vectors, cryptographic key production, and many others.

A deep analysis of randomness and random numbers can be found in [1]. In this paper we will concentrate in the secure design and evaluation of the Random Number Generators as the basic tool for the production of cryptographic keys.

2 Cryptographic security of RNG

In every cryptographic system there are two elements which determine its security, the cryptographic algorithm (which remains constant during the use of the system) and the cryptographic key (which must change in short time periods). For cryptanalytic purposes, the cryptographic algorithm is considered to be known, either because the algorithm from the beginning has been published or it may be compromised during its life time. Therefore, the most critical security element becomes the key, because if the key is compromised to a non authorized person, he can decrypt all the messages that are encrypted with it.

It is obvious that the cryptographic keys must not only have an adequate complexity (big key length) and a short crypto period (frequent key change), but they must have a good quality, which means that they must have sufficient

randomness and unpredictability. The characteristics of the random numbers generators (RNG) make them proper for the production of cryptographic keys. But if the quality of the RNG is not good (low randomness and unpredictability), this could compromise and bypass the total security of a cryptographic system.

Generally, it has to be noted that the cryptanalytic exhaustive key search for the crypto keys can be reduced, if the RNG which produces the keys has less entropy than the key. For example, if a crypto system uses 128 bit keys, which are produced from an PRNG which has only 64 bits seed, then it does not have a key space of 2^128 - as it may first appear - but 2^64.

3 Categories of Random Number Generators

There are two basic categories of Random Number Generators, the True RNG (or Physical RNG) and the Pseudo RNG (or Deterministic RNG). Their basic properties are shown in Table 1.

Table 1: Types of RNG

TRUE RNG (TRNG)	**PSEUDO RNG (PRNG)**
-Non Deterministic RNG	-Deterministic RNG (DRNG)
-Random output	-Pseudo random output
Input Entropy : Physical noise	Input Entropy : Chosen Seed
Implementation : Hardware	Implementation : Software
-Special devices -Hardware failures	-Depend from seed and internal state -Correlations of outputs
-Slower	-Faster

HYBRID RNG
PRNG with regular re-seeding by a true random source

The Hybrid Random Number Generators (HRNG) are mixed RNG, which combine some of the properties of the True and the Pseudo RNG and usually they are implemented by PRNG with a regular re-seeding by a TRNG.

In the following paragraphs we will examine separately the design and security attributes of TRNG , PRNG and HRNG.

4 True Random Number Generators (TRNG)

The TRNG's are based on a physical noise source which produces an analog random signal, which is then digitized and processed to produce random bits in the output, as is shown in Figure 1.

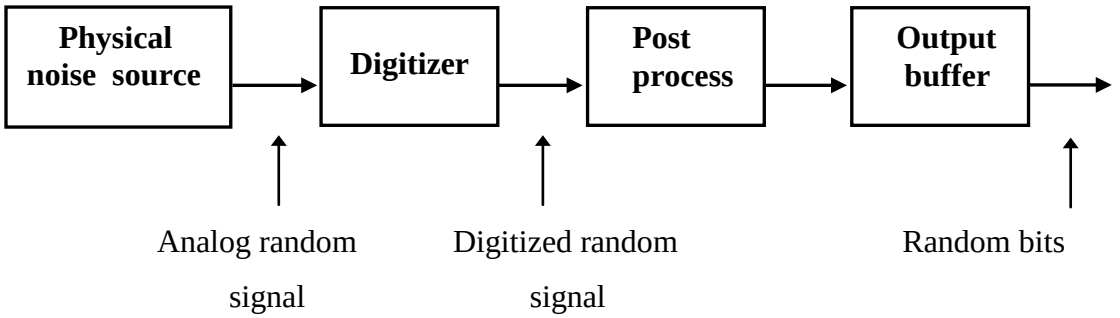


Figure 1: TRNG general diagram

We have to note that regardless of the post process method, the entropy (randomness) of the output bits can not become greater than the entropy of the source which is given by the formula:

$$H(x) = \sum_{i=1}^{n} p(x_i) \log_2[1/\ p(x_i)]$$

where x_i is one of the possible value of the bits (0 or 1) , $p(x_i)$ is the probability of its occurrence and n is the total number of the bits.

4.1. The noise source

The physical noise source in Figure 1 must produce a truly random signal and it can be implemented with one of the following devices:

-Johnson noise (thermal noise in resistors)

-Shot noise (random current fluctuations)

-Avalanche noise (reverse biased Zener diode)

-Free running oscillators

-Quantum optics (reflected or passed photons through a mirror)

-Radioactive sources

In order to reduce the design efforts, the necessary space and the cost, an alternative option for noise source, is the use of some computer internal units which produce enough random noise such as :

-Hard disk chaotic turbulence

-Unplugged microphone input

-Video input (camera lens cap - on)

4.2. Post processing of bits

Since the noise source of a TRNG is random, sometimes it may produce some correlations or uneven distributions between the output bits. As is shown in Figure 1, in order to reduce the correlations and uneven distributions, after the digitization a post processing is applied to the bits. There are many post

processing methods (which are referred also as de-skewing or mixing methods). A detailed description of the above post processing methods can be found in [6]. Bellow we give the more common of them:

1. Mapping the bits with 0 or 1
2. Transition mappings (00 and 11)
3. Stream parity
4. Fast Fourier Transform (FFT)
5. Mixing of two or more inputs (XOR)
6. XOR with LFSR
7. XOR of overlapping bits
8. Compression of bits
9. Hashing of bits
10. Encryption of bits

We must note that the post processing plays an additional security role, because if the physical noise source will fail, the TRNG can work temporarily as a PRNG.

4.3. Evaluation procedures

In order to determine the security level of a TRNG, we must first evaluate the randomness an unpredictability of its output bits and then its total security weaknesses. For this reason the evaluation must be conducted in two different phases:

In Phase 1 the prototype of the TRNG is evaluated against:

1. Secure design
2. Statistical properties (randomness)

In Phase 2 the whole implementation of the TRNG is evaluated against:

1.Tolerances of components
2. Aging of components
3. Temperature limits
5.Malfunction or break down
6.Potential attacks

4.4. Statistical tests for randomness

For the evaluation of the randomness of the output bits, there are a number of statistical tests which can detect abnormalities in their distribution and give an indication of security weaknesses and possible cryptanalytic attacks. Three suites of such statistical tests are shown in Table 2. More details about this subject can be found in [1], [7], [8] and [9].

Table 2: Three suites of statistical test for randomness

NIST SP 800-22	**DIEHARD (Marsaglia)**	**CRYPT-X**
1) Frequency 2) Cumulative Sum 3) Runs 4) Rank 5) Spectral 6) Templates Matching 7) Universal Statistical 8) Approximate Entropy 9) Random Excursions 10) Moving Averages 11) Lempel-Ziv Compression 12) Linear Complexity 13) Bayes	1) Birthday Spacings 2) Overlapping 5-permutation 3) Binary Rank (6x8 Matrices) 4) Binary Rank (31x31& 32x32 Matrices) 5) Monkey tests (20-bit words) 6) Monkey tests (OPSO,OQSO, DNA) 7) Number of 1's in stream of bytes 8) Number of 1's in specific bytes 9) Parking Lot 10) Overlapping Sums 11) Squeeze 12) Minimum Distance 13) Random Sphere's 14) Runs 15) Craps	STREAM CIPHERS 1) Frequency 2) Binary derivatives 3) Change points 4) Runs 5) Sequence complexity 6) Linear complexity BLOCK CIPHERS (1) Frequency (2) Binary Derivative (3) Linear (4) Affine (5) Avalanche (Plaintext) (6) Complementation

There is an open issue, which concerns the overlapping of some statistical tests and a question of how many distinct tests are needed to evaluate the random numbers. In reference [10] there is a short discussion on this issue and an indication for future research.

4.5. Self tests

From the manufacturer's point, the final implementation of the TRNG in order to fulfil the security properties, must incorporate some necessary self tests which must be conducted in the beginning or during its operation and which if they detect some no proper operation or malfunction, they must give a warning to its operator. These self tests must be embedded in the TRNG or exceptionally can be realized externally (by software calling the TRNG). The basic self tests for the TRNG are:

Start up tests : Verify the principle functionality and the randomness of the noise source when the TRNG is started (minimum statistical properties).

On line test : Detect the sufficient quality of noise source, or its deterioration through time.

Total test : Detect the total break down of noise source.

Alarms : In case of weak statistical tests, operational malfunctions and in case that some errors exceed the expected number.

The U.S. standard FIPS 140-2 "Security Requirements for cryptographic modules", [11] , defines four statistical self tests: Monobit test, Poker test, Runs test and Long Run test. According to this standard, these tests must be applied at the start up of the TRNG (or on demand) on a sample output of 20.000 bits. This small sample of the output is selected in order to reduce the time consuming of the statistical tests. But this size of the test sample is not adequate if we want to produce long keystreams (like the case of one time pad cryptographic systems). In

these cases, one must conduct more statistical tests (as these in Table 2) and on bigger output samples.

4.6. Security Classes of TRNG

The U.S. FIPS 140 - 2 defines four Security Levels, which examine the basic security parameters of cryptographic modules. Very briefly, the most important of these parameters are: Ports and interfaces, user authentication, physical security, operational environment, electromagnetic compatibility, key management, self tests on power up or on demand (cryptographic algorithm test, software/firmware integrity test, output statistical tests), design assurance, tampering detection/response and zeroization of the crypto parameters in case of tampering.

Table 3: TRNG application classes

	P1 class (medium strength)	**P2 class (high strength)**
Applications	-Challenges -Initial vectors	- Cryptographic keys and parameters - Random padding - Passwords
Evaluation requirements	- Statistically random output - Self tests during operation	- Stat tests + Entropy tests - Self tests during operation (higher than P1)

The German standard AIS 31 “Functionality classes and evaluation methodology for true (physical) random number generators” [12], defines two

application classes for the TRNG, according to their security strength: The P1 class (medium strength) and the P2 class (high strength). The basic characteristics of these classes are sawn in Table 3. In P1 class, the statistical tests cover only the randomness of the bits and they do not cover their unpredictability. In other words, the statistical tests may detect defects of the random source, but they cannot verify its randomness. For this reason, in P2 class the entropy tests are added (Coron's test, Collision test etc.) which can verify the unpredictability of the bits and thus the randomness of the source.

As is shown in Figure 2, the statistical tests are applied after the post process of the TRNG, but the entropy tests must be done at the output of the digitized signal, because the post process may mask (hide) some dependencies of the bits.

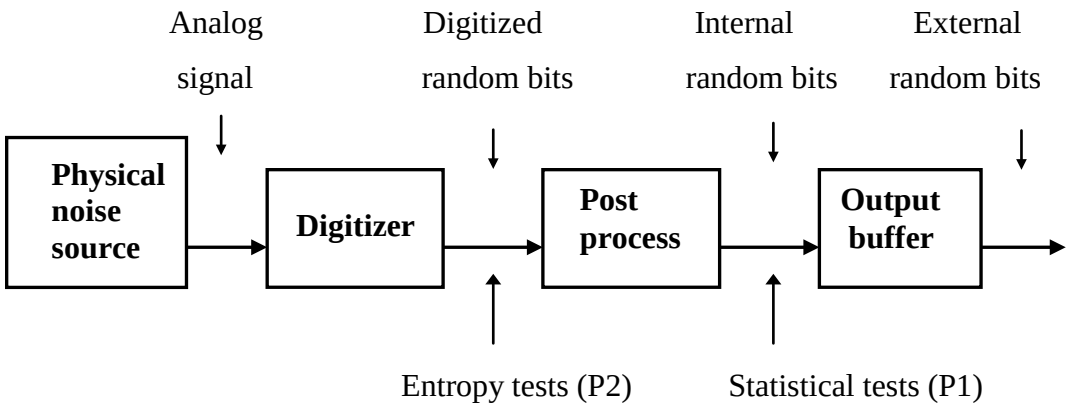


Figure 2: Check points of the Statistical and Entropy tests

4.7. Selection criteria

In addition to the security requirements, there are some other criteria for the selection of the proper TRNG for a cryptographic application, which concerns its functional characteristics and its embodiment into a system. According the application or the total system in which the TRNG has to be integrated, the

priority of these requirements may vary. Reference [4] contains a detailed examination of these functional characteristics. Generally we can briefly summarize the security and functional requirements for a TRNG into the following:

1. Certified randomness
2. Self testing
3. Output rate (speed)
4. Size
5. Embedded or standalone
6. Power requirements
7. Operating temperature
8. Price

5 Pseudo Random Number Generators (PRNG or DRNG)

In a Pseudo Random Number (PRNG) or Deterministic Random Number (DRNG), the source of randomness is not based on a physical process but on a man made predefined process, which is called Seed. And since the Seed is a predefined process (or algorithm), this means that its output is not random, but appears to be random (pseudorandom). As was mentioned in paragraph 3, the need for the use of PRNG is due to their speed and easier implementation in software.

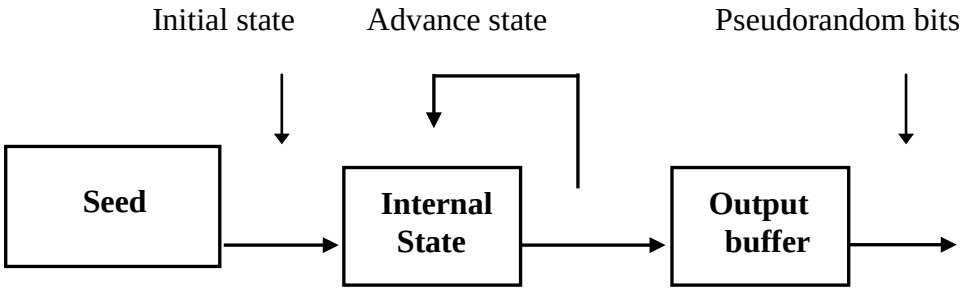


Figure 3: General diagram of PRNG

As is shown in Figure 3, the Seed feeds the Internal State, which is a Finite State Machine (FSM), that can take only certain values. After the Internal State is loaded with an initial state from the seed, it changes its internal state every time it is triggered with a new random number request. This is done with a feedback from its output. (advance state).

We must note that, regardless of any intermediate processing of the bits, the entropy of the output bits can not become greater than the entropy of the seed.

5.1. Entropy of the seed

The Seed in the PRNG must give enough randomness and unpredictability for the initial state. In Table 4 we give some software and hardware sources for the implementation of the Seed, categorized in Low, Medium and High entropy, according to their level of unpredictability.

Table 4: Entropy of different kind of seeds

Low Entropy System Constants	**Medium Entropy** Variable and Unguessable	**High Entropy** External Random
-Configuration files -Drive configuration -Environment strings	-Contents of screen -Computer's date and time -High resolution clock samples -Last key pressed -Log file blocks -Network statistics -Process statistics -Program counter for various Processes	-Cursor position with time -Keystrokes timing -Mouse click timing -Mouse movement -Memory statistics -Microphone input (micr. connected) -Video input

When a PRNG it used for cryptographic key production, there are two important rules that must be followed:

-The entropy of the Seed must be greater than the entropy of the cryptographic Key (which is equal to its length).

-The seed must be frequently updated (re-seeding).

5.2. Design goals for PRNG

The basic design goals for a secure PRNG are given bellow in a priority order:

1. Output indistinguishable from random sequence
2. Knowledge of output does not predict future or past outputs
3. Good use of the entropy in the seed
4. Guaranteed long cycle length
5. Large internal state to avoid exhaustive search
6. Good performance
7. Simple algorithm

5.3. Attacks against the PRNG

A detailed description of the possible attacks against a PRNG is found in [14]. The more important of these attacks are the following:

1. Exhaustive seed search
2. Exhaustive state search
3. Output pre-computation
4. Malicious software attacks
5. Compromise forward tracking
6. Compromise back tracking
7. Control of the input entropy
8. Chosen seed input
9. Cycle shortening
10. Attacks on the operations running time

There are various measures that can be taken against the above attacks, which concern not only the security of the RNG, but the security of the total system which generates and manages the cryptographic keys (computer security). These measures include the isolation of the key generation module (using stand alone computers), the conduction of regular integrity checks (executable code, configuration files etc.) and the hiding of the critical security information of the computer memory [15].

5.4. Application Classes (AIS 20)

The German standard AIS 20 "Functionality classes and evaluation methodology for deterministic random number generators" - ref. [16], defines four application classes for the PRNG's, according to their security strength: The K1, K2, K3 and K4 class. In Table 5 we give their basic characteristics, applications and security requirements.

Table 5: PRNG application classes according to German AIS 20 standard

Class	Apllications	Security Requirements
K1	-Challenges	Mutually different output vectors
K2	-Initial vectors	+ Output with similar statistical properties as ideal random numbers
K3	-Cryptographic keys	+ Minimum bounds for seed entropy + Protection of preceding and following
K4	-Cryptographic keys	+ Protection of preceding outputs, in case of compromised internal state

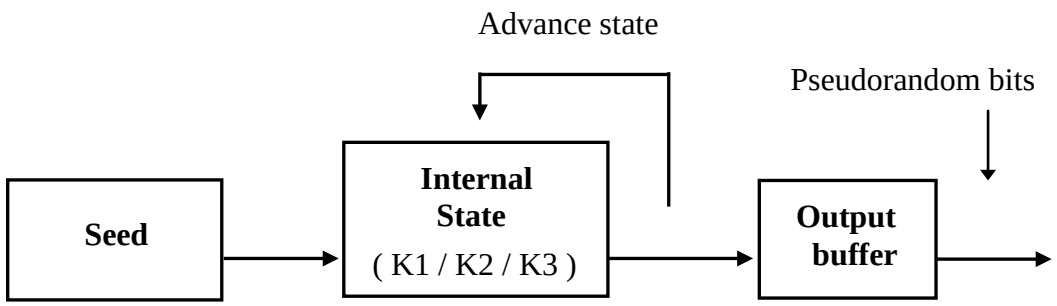
5.4.1 Examples of application classes

The basic component which defines the security class of a PRNG is the Initial State. In Figure 4 we give examples of K1, K2, K3 implementations:

In class K1 the Initial State is implemented by a counter. In this class, the security requirements are only for different output vectors with no statistical properties.

In class K2 the Initial State is implemented by a LFSR or by a block cipher (OFB) with known key. In this class, there is the additional security requirement for output statistics similar to ideal random numbers.

In class K3 the Initial State is implemented by a block cipher (OFB) with secret key. This class, has all the security requirements of the previous classes, with the additional requirement for minimum bounds for seed entropy and for the protection of preceding and following outputs (in case of compromised output).



Class K1 : Counter , Class K2 : LFSR or Block cipher (OFB with known key)
Class K3 : Block cipher (OFB with secret key)

Figure 4: Examples of PRNG application classes K1, K2 and K3 (AIS 20)

In Figure 5, there is another example of K3 class, in which the Initial State is implemented by a LFSR and after this is added an One Way Function (hash function). This prevents an attacker, in the case that he knows some output bits, to calculate preceding or following output bits.

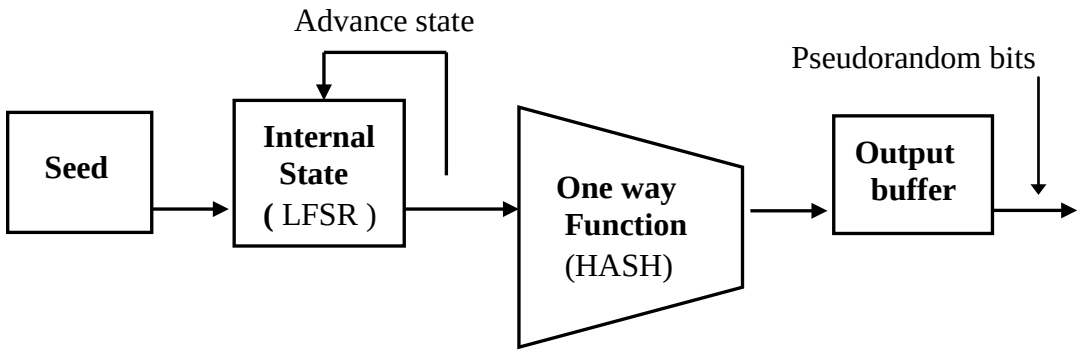


Figure 5: PRNG class K3 implementation with One Way Function

In Figure 6, is shown an example of K4 class, in which the Initial State is implemented by a LFSR with the addition of two One Way Functions, one before and one after the Initial State. The additional One Way Function before the Initial State updates the seed and thus the contents of the Initial State. This not only prevents the calculation of preceding or following output bits (in the case of compromised output bits), but prevents also the calculation of preceding outputs, in the case that some contents of the Internal State are compromised.

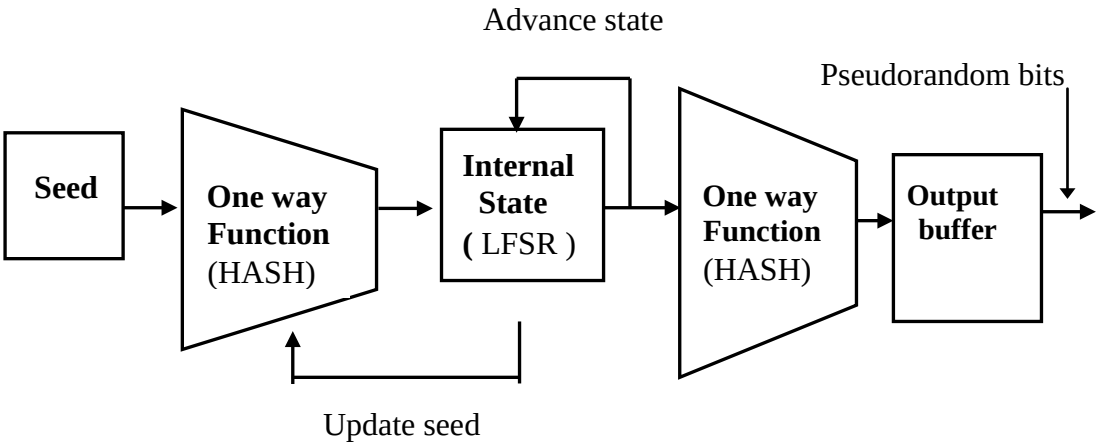


Figure 6: PRNG class K4 implementation with two One Way Functions

6 Hybrid Random Number Generators

The Hybrid Random Number Generators (HRNG) combine some of the properties of True and Pseudo RNG. Their most frequent implementation is the use of a PRNG with a regular re-seeding by a TRNG, as is shown in Figure 7. The substitution of the Seed with a TRNG provides a true random input to the PRNG. This type of HRNG, do not only provide protection against the compromise of the contents of the Initial State (as in the K4 class), but also provides protection against the total control of the Internal State. The Hybrid Random Number Generators provide higher security than the K4 class and belong to the application class H5 (AIS 20 standard).

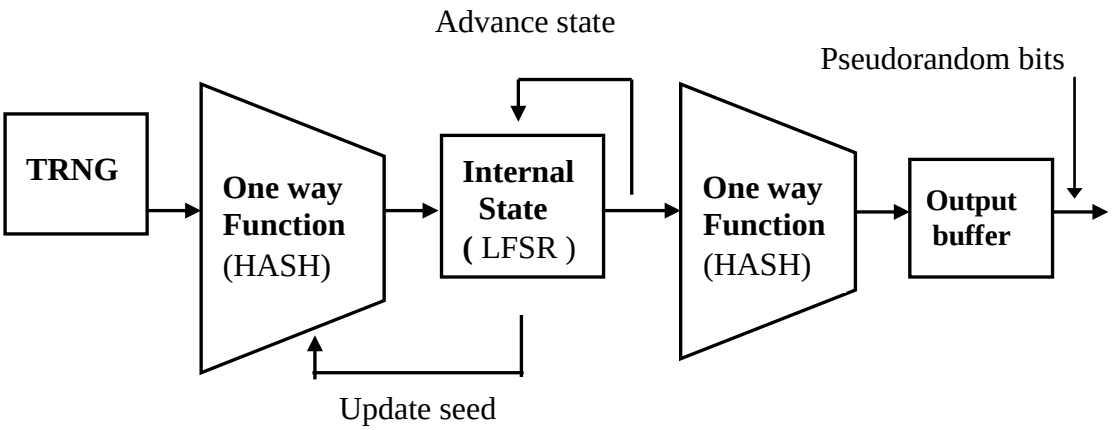


Figure 7: Hybrid RNG (class H5) with a TRNG instead of Seed

Apart from the example of Figure 7, many other types of HRNG can be designed with the combination of TRNG ad PRNG. The more common combination is to mix their outputs with an exclusive OR function. George Marsaglia has proved that the combination of two independent sequences of random numbers, makes the final numbers more evenly distributed than the originals. Also, the combination of two PRNG, provides a longer period to the output bits, which is the least common multiple of the individual periods.

Generally, the combination of RNG's make the output bits more independent, more evenly distributed and harder to predict. A suggestion is to use four types of RNG that at least one of them must be TRNG and the others must be PRNG of different categories [15].

In order to incorporate the various types of hybrid RNG, the AIS 31 standard was updated in 2011, with the addition of new application classes, which mix the TRNG with PRNG [17]. An additional reason for this update, was that if the physical noise source of a TRNG will fail, it can work temporarily as a PRNG, through the post processing (as it was mentioned in paragraph 4.2.).

7 Security evaluation of the total system

As it was mentioned in paragraph 5.3, in order to avoid various attacks, we must not only evaluate the cryptographic security of the RNG, but also the security of the system in which this is embedded. Therefore, as an additional measure we suggest to evaluate the functional security of the total cryptographic key generation system (computer hardware and software), according to the ISO 15408 standard.

The ISO 15408 standard, known also as the Common Criteria for Information Technology security evaluation (CC), is an international set of guidelines, specifications and methods for evaluating the security of the various Information Technology (IT) products. As is shown in Table 6, it provides seven levels of assurance, according to the width of the internal tests and the formality of the security design of the product which is under evaluation. More information on this subject can be found in [18].

Table 6: The Common Criteria (ISO 15408) Evaluation Assurance Levels (EAL)

Assurance Level	Range of Checks
EAL1	Functional Check
EAL2	Structural Check
EAL3	Formal Check and Testing
EAL4	Formal Design, Check and Inspection
EAL5	Semi-officially Design and Check
EAL6	Semi-officially Validated Design and Check
EAL7	Officially Validated Design and Check

8 Additional security measures

If a user knows the basic technical and security aspects of the RNG, he will be able to produce his own trusted cryptographic keys. But many hardware or software cryptographic systems in the market, do not have a Key Input Interface, which will permit the user to load his own trusted keys. Instead of this, they provide an automatic internal key generation unit, in which the key production and the re-keying procedures are not available to the user. The lack of Key Input Interface and external key generation capability, degrade the security of cryptographic systems and decrease the trust of the user in them. Therefore, the users must select cryptographic systems with a Key Input Interface option and they must not use the internal automatic key generation unit, if the key production module is not certified and if the automatic re-keying procedure is not known or not officially approved.

9 Conclusions

We gave a comprehensive introduction of the basic components, the security weaknesses and the design and evaluation methods of deterministic and non-deterministic Random Number Generators. In the following list we summarize the basic security measures which must be taken during the design and the evaluation phase of a RNG, and we suggest some additional measures in order to make it suitable for the production of cryptographic keys:

Measures during the design phase of RNG

1. The Entropy of the Seed must be greater than the Key length (PRNG).
2. Frequent change of Seeds (PRNG).
3. Combined use of TRNG and PRNG (hybrid RNG).
4. Isolation of the Key Generation Module (stand alone computer).
5. Integrity checking (executable code, configuration files etc.).
6. Hide the critical information of computer memory.
7. Mandatory Start Up, On Line and Total statistical and performance tests (TRNG).
8. Alarms in case of weak test results or malfunction (TRNG).
9. Whenever it is possible, obtain random noise from computer hardware.
10. Use cryptographic systems with a Key Input Interface.
11. Use certified internal key generation units with approved re-keying procedures.

Security measures during the evaluation phase of RNG

1. Mandatory Evaluation and Certification of RNG according to International Standards (ISO 15408, AIS 20 & 31, FIPS 140 etc.).
2. Improvement of International Standards for RNG with more on line tests on bigger samples.
3. Research on the minimum required number of distinct and independent statistical tests.

4. For higher levels of security, in addition to statistical tests, conduct entropy tests to detect unequal distributions and dependencies.
5. Check not only the design, but also the concrete implementation of RNG , as well as the total system in which this is embedded.

References

[1] Donald Knuth, *The Art of Computer Programming-Volume 2/ Seminumerical Algorithms,* Addison-Wesley 1998.

[2] Bruce Schneier, *Applied Cryptography*, John Wiley, New York, 1996.

[3] Alfred Menezes, Paul C. van Oorschot, Scott A.Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[4] M. Dugan, *Analysis of existing implementations of TRNG*, May 2005.

[5] Carl Ellison, *P1363:Appendix E-Cryptographic Random Numbers, Draft V1.0,* November 1995.

[6] D. Eastlake, S. Crocker and J. Schiller, *RFC 1750 - Randomness Recommendations for Security*, Network Working Group, December 1994.

[7] NIST Special Publication 800-22, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,* National Institute of Standards and Technology (NIST), April 2010.

[8] Helen Gustafson et. al., A computer package for measuring strength of encryption algorithms, *Journal of Computers & Security*, **13**(8), (1994), 687-697.

[9] G. Marsaglia and G. Diehard, *Battery of Tests of Randomness*. Available at: http://stat.fsu.edu/pub/diehard/ (1996), http://stat.fsu.edu/pub/diehard/

[10] Juan Soto, *Statistical Testing of Random Number Generators*, National Institute of Standards and Technology (NIST).

[11] FIPS 140-2, *Security Requirements for cryptographic modules*, National Institute of Standards and Technology (NIST), May 2001.

[12] W. Killmann and W. Schindler, AIS 31: *Functionality classes and evaluation methodology for true (physical) random number generators*, version 3.1., Bundesamt fur Sicherheit in der Informationstechnik (BSI), Bonn 2001.

[13] Tim Matthews, *Suggestions for Random Number Generators in software*, RSA Data Security Engineering Report, December 1995.

[14] Robert W. Baldwin, Preliminary Analysis of the BSAFE 3.x Pseudorandom Number Generators, *RSA Laboratories Technical Bulletin*, **8**, (Sept., 1998).

[15] W.W. Tsang, C.W. Tso, Lucas Hui, K.P. Chow, K.H. Pun, C.F. Chong and H.W. Chan, *Development of Cryptographic Random Number Generators*, Department of Computer Science and Information Systems, University of Hong Kong, August 2003.

[16] AIS 20, *Functionality classes and evaluation methodology for deterministic random number generators*, Bundesamt fur Sicherheit in der Informationstechnik (BSI), December 1999.

[17] Viktor Fischer, A Closer Look at Security in Random Number Generators Design, Laboratoire Hubert Curien, Jean Monnet University, May 2012.

[18] https://www.commoncriteriaportal.org/