Volume XXIII 2020
ISSUE no.1
MBNA Publishing House Constanta 2020

Scientific Bulletin of Naval Academy

SBNA PAPER • OPEN ACCESS

An overview of hybrid random number generators

To cite this article: Veronica Cornaciu and Ciprian Răcuciu, Scientific Bulletin of Naval Academy, Vol. XXIII 2020, pg.248-252.

Available online at www.anmb.ro

ISSN: 2392-8956; ISSN-L: 1454-864X

doi: 10.21279/1454-864X-20-I1-034

An overview of hybrid random number generators

V Cornaciu[1] and C Răcuciu[2]
[1]PhD Candidate, Military Technical Academy-Electronic, Information and Communications Systems for Defense and Security, George Coșbuc nr. 39-49, Bucharest, 050141, Romania.
[2]Prof. Eng. PhD, Titu Maiorescu University-Faculty of Computer Science, Văcărești nr. 187, Bucharest, 004051, Romania.
[1]veronica_zanfir@yahoo.com
[2]ciprian.racuciu@gmail.com

Abstract. The generation of random numbers is a important topic in cryptography. Random number generators are bradly divided into two categories: random number generators(RNGs) and pseudo-random number generators(PRNGs). If the PRNGs werw intensively studied in the specialized literature, many such generators being built and analyzed, the topic of RNGs did not capture the researchers atention so much. Candidates in this first category generate non-deterministic sequences and are often based on physical reactions, such as radioactive degradation or mouse movement. A special category of generators is the one that combines the two categories, namely, the category of hybrid generators (HRGs). The purpose of this paper is to study in detail the category of hybrid generators and to provide a detailed analysis of the results of statistical tests, security , portability and how to improve some of the generators of this category.
Keywords: Hybrid random generators, statistical test, portability, security.
The first section in your paper

1. Introduction

The generators of random numbers and generators of random bits, the RNG and RBGs, are a fundamental tool in many different domains. Numbers obtained with these generators are used in statistical sampling, cryptography, simulation, gambling, etc. Pure random numbers guaranty the integrity of cryptographic protocols used in secure communications and the accuracy of Monte-Carlo simulations used in biomedical sciences . No matter the domain they are used, random numbers generators have to accomplish some essential conditions with a higher or lower weight. These must pass the main statistical tests, must be robust from the cryptographic point of view, to have a relatively high generation speed and to generate a large quantity of random numbers. There are numerous studies in which different generators are analysed, but only a few that propose a selection method of them. In [1], authors developed a multi-criteria evaluation procedure of pseudo-number generators to weigh them in relation to others and a method of determining the weight of each evaluation criterion.

In the context of random numbers, the notions "real random numbers" and "true random number generators" (TRNGs), appear quite frequent. By real random number is understood the independent realisation of a random variable uniform distributed and by TRNGs we note the generators which result from physical experiment considered to be random, like the radioactive degradation or the noise from a semiconductor diode.

If the random pseudo-numbers are obtained from deterministic math algorithms, these, in order to be pure, can be extracted from physic entropy sources, like electrical noise or quantum fluctuations. These types of generators extract and store data from the source then try to estimate the entrance entropy and limit the number of exit bits at the quantity of entropy gathered. We can name these generators *Entropy collector generators.* The quality of the entropy collector generators depends in a large measure on the selection of noise sources and on the adequate character of entropy estimation. There also exists another category of random number generators, a category that combines the two variants above, this being the category of hybrid generators. These, not only that collect the entropy, but also use a few additional methods to raise the quantity of random numbers generated.

The present work is dedicated to the study of hybrid generators. The article is organized as follows: Section 2 presents the main techniques used in hybrid generators, focusing on the structure of the Yarrow and HAVAGE generators. In section 3 a comparative study of the main parameters of the two generators is made. The article ends with the conclusions.

2. Techniques used in hybrid generators

Some systems use a hybrid method to generate random numbers. Initially it offers randomness from natural sources, and then, when these sources are not available, they return to pseudo random numbers generated by generators of secure random numbers from a cryptographic point of view(CSPRNGs). The switching is performed when the ability to generate random numbers does not keep up with the demand. This method avoids blocking the generation of numbers in the case of slower generators or for purely environmental causes.

A hybrid random number generator (HRNG) includes an output, a combinational logic, a TRNG, and a PRNG. The HRNG is configurable to operate in a first and a second mode, wherein in the first mode the PRNG is serially connected between the TRNG and the output and the TRNG intermittently influences the PRNG, and in the second mode the TRNG and the PRNG are connected to the output via the combinational logic.

Some of the most eloquent generators in this category are Yarrow and HAVAGE.

Yarrow constantly collects entropy from external sources and if the user requests more random numbers than the estimated value of collected entropy, an additional PRNG is used to extend the collected information.

On the other hand, HAVEGE is following another strategy. It collects the input from a noise source and amplifies the entropy collected by directly influencing the source. The "chaotic" behavior of the processor is enhanced by two random walks in a table with random numbers.

Yarrow was developed at Counterpane Systems by N. Fergueson, J. Kelsey and B. Schneier [2] and was designed to prevent certain types of attacks described in [3] and to ensure high performance and portability. Also, through its structure it is allowed to adjust each component of the generator according to the user's needs.

Yarrow is made up of two separate pools, a fast pool *Pf* and a slow pool *Ps*. A cryptographic hash function is used to feed the input into the two pools. When sufficient entropy is collected in the two basins, then a secret key *K* is generated for a block cipher of their contents. Using an encryption function and the secret key random numbers are generated.

The Yarrow concept consists of four main components: Entropy Accumulator - responsible for processing the input, mixing it in the two basins and estimating the collected entropy, Reseed Mechanism - if the basins are in an unpredictable state it generates a new key from their contents, Reseed Control - initiates a re-seeding of the key if sufficient entropy is collected in the basins, Generation Mechanism - applies a block cipher and secret key to produce random numbers and periodically changes the key, a process that is carried by the generator (generator gate).

HAVAGE (Hardware Volatile Entropy Gathering and Expansion) produces random numbers using uncertainties that arise in processor behavior after a break. This generator was developed by Sendrier and Seznec at INRIA (Paris, FR) and is described in [4] and [5]. There are two variants of this generator, namely HAVEG (Hardware Volatile Entropy Gathering) that gathers passive entropy and HAVEGE.

HAVEGE consists of a table, the so-called Walk-Table and two indicators PT and PT2 that represent the current position of two simultaneous movements within the table. In the beginning, the table receives random numbers generated by HAVEG, after which, the transition function deals with the continuous renewal of the numbers in this table. The course of the two rides is determined by the content of the table, its size being chosen to be twice as large as the data L1 cache. Therefore, the probability is about 50% that the data from the sheet accessed by the two rides will be cached.

The output function combines only the current values of the two rides. Since the content of the table and thus the course of the two rides can be considered random, this simple mechanism is sufficient to generate good random numbers.

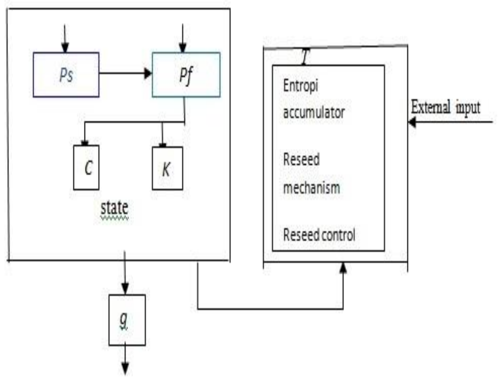

Figure 1: General structure of Yarrow

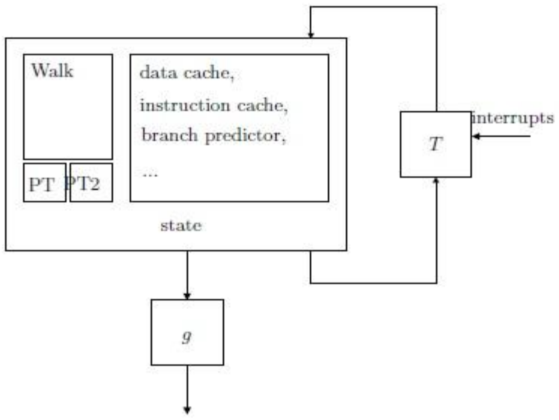

Figure 2: General structure of HAVAGE

3. Comparative study of the characteristic parameters

3.1. Internal state

The internal state of Yarrow depends on the two pools, *Pf* and *Ps*, on the secret key *K* and the counter C. When the estimated entropy of the pools exceeds a certain threshold, *K* and *C* are displaced using the two pools. The fast pool is used for re-seeding the key, and the slow one ensures the quality of the reseeding. In this way, the contents of the pools cannot be guessed. Also, a block cipher is used to encrypt the counter value and then random numbers are generated.

After a basin has been used for reseeding, its entropy is reset to zero. The entrance is mixed alternately in the two pools. In this way, the content of a pool is represented by the hash value of the previous entry, mixed with its content. Using a hash function together with a block cipher, if the hash function had an output of *m*-bits and the block cipher would use a key on *k*-bits, then the generator power is limited to *min (m, k).*

As for HAVEGE, the internal state consists not only of the Walk-Table and the two indicators, but also of all the volatile states inside the processor, which determines the number of clock cycles required. For this reason, the entire state of HAVEGE can only be observed if the processor clock is frozen, which is impossible. The Walk-Table size was chosen to be twice the size of the L1 cache.

Also, the power of the generator also depends on the volatile states of the processor because they are part of the internal state of the generator, so they must be taken into consideration. In other words, HAVEGE's power is given by the size of the Walk-Table and the thousands of volatile states.

Initially Walk-Table is seeded by HAVEG using the HARDTICK () method, which measures the number of hardware clock cycles since the last call of this method. The call of HARDTICK () is embedded in a small sequence of instructions, which uses a read and write command, as well as a conditional branch. The HARDTICK () output is included in position *K* in the Entrop-Array by means of a simple mixing function, a function consisting of cyclic shift and XOR operations. The change operation is compensated for the shortcomings related to the fact that the largest amount of entropy is found in the least significant bits and the XOR-has operations are used to combine the input value with the old values from position *K* and *K +1* from table. Thus, each input value has an influence on each position in the array, after an appropriate time.

The disadvantage of this program is that, if no interruptions occur, the algorithm becomes extremely deterministic, and the output contains only a small entropy.

3.2. The transition function

The biggest shortcoming of RNGs is the overestimation of entropy. For this reason, Yarrow uses the following strategy. The amount of entropy of each source and each pool is estimated, recorded separately and reset to zero after each seeding. Because different counters are used, if the entropy of a source is overestimated, the entire entropy estimation is not endangered. The reseed control uses the entropy counter of the pools to determine the moment of new reseeding of the key and the fast pools is responsible for the frequent reseeding when the accumulated entropy reaches a certain threshold. For this reason, the system recovers quickly after a key has been compromised. The slow pool has the role of ensuring the quality of the reseeding of the key and is used only if at least one source exceeds the threshold chosen by the amount of entropy collected. The *r* number can thus be adjusted according to the chosen environment.

In the case of HAVEGE, the transition function uses two simultaneous walks through the table. The table is twice as large as the data L1 cache. For this reason, the probability of an access to the table, due to missing the cache is about 50%.

Because the Walk-Table is full of random numbers, the behavior of the processor is changed in a random way. In the absence of interruptions, HAVEGE would become a purely deterministic algorithm, but using the table helps to compensate for the interruption deficiencies.

The two simultaneous walks are represented by the two pointers PT and PT2. PT is influenced by its current position, table content and HARDTICK () result. The new PT value is used to refresh the contents of a cell in the sheet. PT2 is influenced by its current position, table content and PT value. Thus, the HARDTICK () result has a direct and indirect influence on the behavior of both rides. If an opponent finds out the contents of the table, as well as the two pointers, the unknown result of a new processed HARDTICK() makes this knowledge unnecessary. Also, the two indicators, which represent the current position of the two walks inside the table, are combined with XOR and placed in the output buffer. In this way, the opponent is prevented from guessing the contents of the Walk-Table when monitoring the generator output.

3.3. Portability

Due to the fact that Yarrow is a concept of HRNgs, it can be adjusted to be implemented on any platform and with the specific requirements of the user. The generator has several implementation variants. Thus, Yarrow-160 is implemented in WinNT and Win95, but can also be implemented in Mac OS X. As a simple operator-operated program, HAVAGE does not use call operating systems. It can also be used with machines and operating systems that use optimization techniques such as caches or branch predictors. In these conditions, it is sufficient to adjust the specific parameters, in particular the size of the branch predictor, the L1 cache of the data, the number of code iterations, the size of the Walk-Table, etc., in order to run the program.

3.4. Security

Yarrow has been specially designed to prevent all attacks described in [3]. The separation of the pools and the key, causes the input samples to have no intermediate influence for the output. In this way iterative guess attacks are avoided. The fast pool allows the generator to recover quickly after a compromised key, the slow pool prevents damage caused by overestimating the entropy. The generator gate limits the number of output bits that can be used for a backtracking attack. Furthermore, Yarrow uses cipher blocks and cryptographic hash functions, so it has high resistance to collision attacks. keys, causes the input samples to have no intermediate influence for the output. In this way iterative guess attacks are avoided. The fast pool allows the generator to recover quickly after a compromised key, the slow pool prevents damage caused by overestimating the entropy. The generator gate limits the number of output bits that can be used for a backtracking attack. Furthermore, Yarrow uses cipher blocks and cryptographic hash functions, so it has high resistance to collision attacks.

The weak points of the generator are the power of generation and the bad input samples. Thus, the power of the Yarrow-160 generator is only 160 bits, which could be improved by using functions with a larger hash and a larger key-block cipher. Also care must be taken when selecting input sources.

HAVEGE's security is based on the unobservable internal state of the generator. If an opponent tries to read the internal state of the generator, an interruption occurs and thus the internal state is modified. The only option would be to freeze the hardware clock, but this is only possible in special laboratories.

HAVEGE resets the internal state permanently. Therefore, even if an opponent memorizes the contents of the running board and the two indicators, he loses track of the moves and therefore of the future random numbers as soon as the next HARDTICK result is processed. This means that the generator is able to recover very quickly from a compromised state.

Conclusion

In this paper we examined two of the most eloquent generators in the hybrid generators category. The two generators are examined from the perspective of the general structure, the internal state, the portability and the security they benefit from. Although Yarrow has been replaced by its improved Fortuna [6], it remains a generator model that can provide data and ideas for building new generators.

Yarrow's individual components can be easily replaced and adjusted according to the client's wishes. In [1] particular elements of the generator and their behavior are analyzed during the different types of attacks, but there is no data of the empirical tests or about the efficiency of the specific implementations. Yarrow is a simple and robust example of combining the entropy collected with a PRNG, the counter and the block digit, being nothing more than a PRNG.

HAVAGE is a modern concept that uses the entropy that is generated from the unpredictable behavior of the processor. Regarding the production of random numbers but also the speed of generation, the quality of this generator is comparable to that of a pseudo-random number generator. The safety of the generator is quite high precisely because the internal state monitoring does not help in case of an attack because any attempt to modify it will reset the generator.

References

[1] Cornaciu V and Răcuciu C 2019 Multi-criteria method for evaluation of the pseudorandom number generators using thermodynamic systems behaviour *Mircea cel Batran Naval Academy Scientific Bulletin* **22(2)** pp. 305-312

[2] Kelsey J, Schneier B and Ferguson N 2001 Yarrow-160: Notes on the design and analysis of the yarrow cryptographic pseudorandom number generator *International Workshop on Selected Areas in Cryptography* pp. 13-33

[3] Kelsey J, Schneier B, Wagner D and Hall C 1998 Cryptanalytic attacks on pseudorandom number generators *Lecture Notes in Computer Science* **1372** pp.168-188

[4] Sendrier N and Seznec A 2002 HArdware Volatile Entropy Gathering and Expansion: generating unpredictable random numbers at user level **RR-4592** INRIA

[5] Sendrier N and Seznec A 2003 HAVEGE: A user-level software heuristic for generating empirically strong random numbers *ACM Transaction on Modeling and Computer Simulations* **13(4)** pp.334-346

[6] Schneier B and Ferguson N 2003 Practical Cryptography *John Wiley & Sons.*