A HTTP-Enabled Cryptographically Secure Hardware Random Number Generator

Mislav Božičević * and Aleksander Radovan **
* Polytechnic of Zagreb, Zagreb, Croatia, mb@mail.mislav-bozicevic.iz.hr
** University of Applied Sciences Velika Gorica, Velika Gorica, Croatia, aleksander.radovan@vvg.hr

Abstract - A cryptographically secure hardware HTTP enabled random number generator based on a nondeterministic entropy source has been implemented using open source hardware development kit. Internally, randomness is distilled from the entropy source using Algorithm M where modified-RC4-based and BLAKE-512-as-counter-mode pseudo-random number stream generators are combined. Users access the device via HTTP on a TCP/IP network and platform specific device drivers are not required. The random number files were tested using stringent statistical tests including DIEHARD, STS 2.1.1, dieharder 3.31.1, TestU01 1.2.3 SmallCrush and ENT. Emphasis was put on low cost of hardware.

I. INTRODUCTION

From the very beginnings of the computer systems, random numbers played an important role. John Von Neumann, who is regarded as the father of computer science, strongly cautioned people against the use of any form of software algorithm to generate random numbers [7]. Von Neumann recognized that only a physical process can produce a truly random, unpredictable number. The output of a mathematical algorithm, by its very nature, can be predicted if the algorithm is returned to the initial condition of previous time.

A good random number generator should have several characteristics in order to satisfy strict mathematical requirements of randomness:

- It should produce numbers that pass major known criteria for randomness, like statistical tests including DIEHARD, STS 2.1.1., dieharder 3.31.1, TestU01 1.2.3 SmallCrush and ENT.
- It should not have repetitive pattern in its output.
- It should have no initial conditions that reset the output to repeat a sequence.
- It should not be spoofable.
- It should not be susceptible to commandeering, in which the generator is forced to produce a predictable stream.

This work was supported by the University of Applied Sciences Velika Gorica and the Polytechnic of Zagreb.

Unlike pseudo-random number generators, which can be placed in a predictable state by introducing a "seed" number into the algorithm, hardware-based random number generator represents a powerful alternative.

Hardware random number generator generates random numbers from a physical process, rather than a computer program. It consists of a device that converts physical phenomena to a signal that is converted to the output digital number. Series of such values can be obtained by repeatedly sampling the input signal.

In order to be maximally utilized in a network environment, hardware random number generator must be able to connect to different devices via USB or RJ45 interface. This example shows an implementation of hardware number generator using nondeterministic entropy source like zener diode or avalanche photodiode, LeafLabs Maple (Rev5) board with microcontroller unit and network module Wiznet WIZ820io. Collected random number data stored in files were tested using stringent statistical tests that proved the truly randomness of the numbers.

II. RANDOM NUMBER GENERATION

A. Block diagram

Microcontroller unit samples a nondeterministic entropy source using an analog-to-digital converter. Samples are stored internally at the MCU in the entropy pool which is cryptographically transformed in an irreversible manner. Cryptographically processed, the numbers are available to the user via HTTP over TCP/IP.

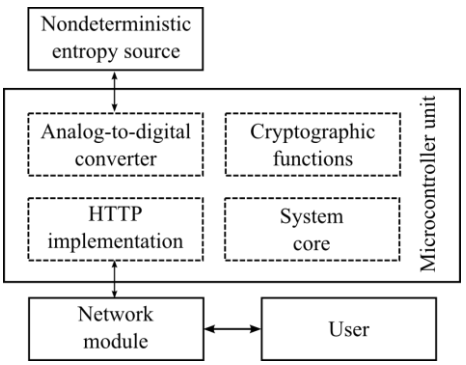


Figure 1. Block diagram that shows connections between the nondeterministic entropy source, the microcontroller unit and the network module

The nondeterministic entropy source is connected to the analog-to-digital converter pin of the MCU. The network module is connected via SPI with the MCU. The user connects to the device via HTTP over a TCP/IP network (as shown in Figure 1).

B. Microcontroller unit

Maple (Rev5), open source microcontroller board, was the basis of the hardware number generator. The board is based on STM32F103RB microcontroller unit ARM Cortex-M3 32-bit RISC core operating at a 72 MHz frequency capable of handling the input, transformation and output tasks. It is equipped with 2 x 12-bit A/D converters, which were used for processing the analog data collected from nondeterministic entropy source.

The board offers custom programming capabilities through uploading new code onto it using USB interface, which is directly supported by the Cortex-M3 MCU. This option was used for customizing the data processing and communication with the Network module WIZ820io.

Our HTTP implementation includes the GET and HEAD methods according to the HTTP/1.1 specification [14]. Individual resources available to the client are fixed length sequences of cryptographically processed numbers. Client may use the GET method to obtain an index of available resources by requesting the root directory from the server, e.g., http://192.168.0.2/. When an individual resource is requested, its URI (for example http://192.168.0.2/1234BEEF) is removed from the index and another URI is put in its place (for example http://192.168.0.2/1234BEF0). Number of URIs in the index, and therefore the number of parallel users (client connections), is limited by MCU memory and network module capability. HEAD method is provided for the client to check if there were changes on the index. HTTP was chosen for its widespread use, but simple modifications in the source code would allow for custom protocol, e.g., UDP-based.

C. Network module

WIZ820io is an internet offload network module that includes W5200, MAG-JACK (RJ45 with transformer) and other glue logic. W5200 is at the core of WIZ820io as a hard-wired TCP/IP chip which includes an Ethernet PHY. The W5200 is TCP/IP embedded Ethernet controller that enables internet connection for embedded systems using SPI. It offers eight sockets to programmer to be used for a web browser, Telnet server or protocols like TCP/IP or UDP.

Connection between Maple board and WIZ820io module was set up by using WizEthernet library and by connecting pins “RESET”, “LOW POWER MODE”, “INTERRUPT” and SPI bus.

D. Nondeterministic entropy source

Hardware random number generators employ a physical process to produce random numbers. The specific physical processes employed to generate random numbers in many cases include thermal noise or quantum processes.

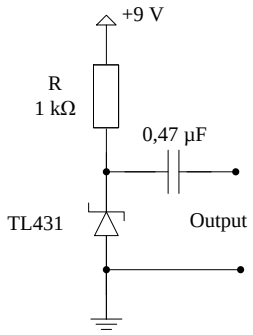


Figure 2. Zener diode based thermal noise source circuit

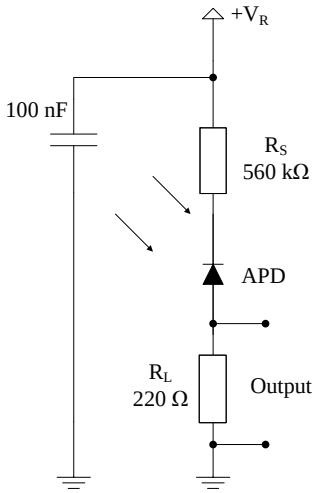


Figure 3. Simple passive quenching circuit for avalanche photodiode SAP500S2

Thermal noise arises from the agitation of current carriers due to environment temperature effects and is independent of external voltage applied to the device. Thermal noise is an example of white noise and does not have its origins in a quantum mechanical process, but rather in stochastic processes governed by the laws of thermodynamics.

One of the implementation of thermal noise is avalanche noise in which the breakdown noise generated by a zener diode operating in reverse bias mode is measured (circuit show in Figure 2). Zener diode was one of the nondeterministic entropy source that was used in the described random number generator.

Other nondeterministic entropy source that can be used in random number generator is avalanche photodiode (APD) that works in the “Geiger” mode to allow detection of single photons. One of these avalanche photodiodes is Laser Components SAP500S2. To allow detection of single photons, the SAP500S2 can be biased above the breakdown voltage using the passive quenching circuit (as shown in Figure 3).

TABLE 1. SELECTED DIEHARDER TEST RESULTS

test_name	tsamples	p-value
diehard_birthdays	100	0.84098386
diehard_operm5	1000000	0.45785809
diehard_rank_32x32	40000	0.12351172
diehard_rank_6x8	100000	0.99121283
diehard_bitstream	2097152	0.31022947
diehard_opso	2097152	0.06986824
diehard_oqso	2097152	0.58840360
diehard_dna	2097152	0.02095990
diehard_count_1s_str	256000	0.75738672
diehard_count_1s_byt	256000	0.42087518
diehard_parking_lot	12000	0.28849924
diehard_2dsphere	8000	0.98750858
diehard_3dsphere	4000	0.04579137
diehard_squeeze	100000	0.90795986
diehard_sums	100	0.16653095
diehard_runs	100000	0.96566529
diehard_craps	200000	0.15539040
marsaglia_tsang_gcd	10000000	0.33710278
sts_monobit	100000	0.93145119
sts_runs	100000	0.94482231
rgb_kstest_test	10000	0.94587621
dab_bytedistrib	51200000	0.85997660
dab_dct	50000	0.30514859
rgb_permutations	100000	0.00407979

III. CRYPTOGRAPHIC DISTILLATION

Our cryptographic building blocks are modified RC4 [6] stream cipher and BLAKE-512 [1] hash function. Only modification on RC4 is that it skips its first 3328 PRGA outputs. That number of skipped outputs is more conservative than the most conservative recommendation of $12 \cdot 256$ skipped outputs [3, p.14].

A PRNG is constructed from BLAKE-512 [1, p.27]. RC4 PRGA is already a PRNG seeded by its KSA.

Algorithm M (Randomizing by shuffling) [4, p.33] is used to combine the two mentioned PRNGs.

Our setup consists of two memory buffers. One is an analog-to-digital converters buffer where signal from the nondeterministic entropy source is digitized and stored. The other is a randomness pool where the hash value (computed using BLAKE-512 algorithm) from the concatenation of the previous randomness pool contents and the entire analog-to-digital converters buffer is stored.

When the analog-to-digital converters buffer is exhausted, it is replenished by new measurements from the nondeterministic entropy source and the randomness pool is updated as previously described.

Random bytes are extracted from the randomness pool by using BLAKE-512 as a counter-mode pseudo-random byte-stream generator similarly to [5, p.427].

Random bytes extracted in that fashion are used as keys for the modified RC4 and to fill the delay buffer for Algorithm M. Algorithm M uses RC4 PRGA to derive an index to the delay buffer. Value from the delay buffer (selected by the index) is served to the user. After that value is used, it is updated by XOR operation with the next sample from the analog-to-digital converters buffer.

IV. STATISTICAL ANALYSIS

Currently, there is no general definition of randomness [9] and no finite number of randomness tests can prove that a sequence of numbers is random; they can only prove that the sequence is not random. Often, a sequence (of bits) is considered random if and only if it is shorter than any computer program that can produce that string.

A. *ENT*

To check the randomness of our cryptographically processed sample, the widely used program ENT was applied to calculate entropy, Chi square distribution, arithmetic mean, Monte Carlo estimation of Pi and serial correlation coefficient. The test results for a typical sequence of 1574397 bytes are as follows:

Entropy = 7.999727 bits per byte.

Optimum compression would reduce the size of this 1574397 byte file by 0 percent.

Chi square distribution for 1574397 samples is 595.11, and randomly would exceed this value less than 0.01 percent of times.

Arithmetic mean value of data bytes is 127.5446 (127.5 = random).

Monte Carlo value for Pi is 3.136277196 (error 0.17 percent).

Serial correlation coefficient is -0.000708 (totally uncorrelated = 0.0).

B. *Dieharder*

The sample passed all tests in the dieharder battery, except one which was marked as a weak pass (rgb_permutations) as summarized in table 1. Dieharder is a random number generator testing suite [12].

C. TestU01 SmallCrush

TestU01 1.2.3 SmallCrush battery of tests was applied and the results are summarized in table 2. TestU01 is a software library implemented in ANSI C, and offering a collection of utilities for empirical statistical testing of uniform random number generators [10].

D. STS

The sample passed 183 of 187 tests in the STS 2.1.1 battery of tests [13]. Among those failed the lowest grade (as proportion) was 87/99 in the ApproximateEntropy test. The other three failed tests (all are NonOverlappingTemplate tests) share the same grade (94/99).

STS 2.1.1 was carried out with the default parameters and the sample was interpreted as 99 sequences 15903 bytes each.

E. DIEHARD

DIEHARD battery of tests was carried out and compared with dieharder results. The results match as expected.

Most of the tests in DIEHARD return a uniform value on [0,1), which is named "p-value", if the input data contains fully independent random numbers. Those values are calculated using assumed distribution [11].

V. Practical Applications and Uses

Hardware random number generators can be used for generating security keys for SSL protocols, generating initialization vectors for a symmetric algorithm for encryption of files, generating cryptographic primitives like *nonce* (number used once) in the Kerberos authentication network protocol, generating asymmetric keys used in PKI, etc.

VI. Conclusion

Results from the empirical type of statistical tests here presented gives strong support to the expectation that our hardware random number generator could be used in a real-world scenario. Cost of hardware was lowered by selecting modular components which are open source hardware and no platform dependency was introduced by using HTTP over the widely available TCP/IP.

The security of the cryptographic transform was built on the thoroughly studied RC4 and BLAKE-512. On a carefully administrated internal network our device can serve several simultaneous users.

There is space for improvement in customization of statistical tests to better suit the cryptographic construct parameters.

TABLE 2. SUMMARY RESULTS OF SMALLCRUSH

test_name	pvalue
Collision	< 1.0e-300
Gap	< 1.0e-300
SimpPoker	< 1.0e-300
CouponCollector	< 1.0e-300
MaxOft	< 1.0e-300
MaxOft AD	1 – (<1.0e-15)
WeightDistrib	< 1.0e-300
HammingIndep	< 1.0e-300
RandomWalk1 H	< 1.0e-300
RandomWalk1 M	< 1.0e-300
RandomWalk1 J	< 1.0e-300
RandomWalk1 R	< 1.0e-300
RandomWalk1 C	< 1.0e-300
MatrixRank	0.86
BirthdaySpacings	0.61

TABLE 3. STS RESULTS

Statistical test	Proportion	p-value
Frequency	99/99	0.246470
BlockFrequency	97/99	0.458035
CumulativeSums	99/99	0.637119
CumulativeSums	98/99	0.341565
Runs	99/99	0.613995
LongestRun	96/99	0.012650
Rank	98/99	0.000004
FFT	98/99	0.834308
Universal	99/99	0.000000
ApproximateEntropy	87/99	0.000000
Serial	98/99	0.834308
Serial	99/99	0.660243
LinearComplexity	96/99	0.522989

List of Abbreviations

APD	…	Avalanche photodiode
HTTP	…	Hypertext Transfer Protocol
IP	…	Internet Protocol
KSA	…	Key-scheduling algorithm
MCU	…	Microcontroller Unit
OSI	…	Open Systems Interconnection
PHY	…	Physical Layer of the OSI

		Model
PKI	…	Public Key Infrastructure
PRGA	…	Pseudo-random generation algorithm
PRNG	…	Pseudorandom number generator
SPI	…	Serial Peripheral Interface Bus
SSL	…	Secure Socket Layer
TCP	…	Transmission Control Protocol
UDP	…	User Datagram Protocol
URI	…	Uniform resource identifier

ACKNOWLEDGEMENTS

The authors are grateful to University of Applied Sciences Velika Gorica for financial support for avalanche photodiodes SAP500S2, to Polytechnic of Zagreb for financial support and to Professor M. Stipčević from the Ruđer Bošković Institute for helpful advices.

REFERENCES

[1] J.-P. Aumasson, L. Henzen, W. Meier, C.-W. R. Phan, "SHA-3 Proposal BLAKE version 1.3", available at https://131002.net/blake/

[2] A. Dujella, M. Maretić, "Kriptografija", Element Zagreb, 2007.

[3] I. Mironov, "(Not So) Random Shuffles of RC4", in Cryptology ePrint Archive, Report 2002/067, http://eprint.iacr.org/, 2002.

[4] E. D. Knuth, "The Art of Computer Programming: Volume 2 / Seminumerical algorithms Third Edition", Addison-Wesley, 1997.

[5] B. Schneier, "Applied Cryptography Second Edition : protocols, algorithms, and source code in C", John Wiley and Sons, Inc, 1996.

[6] Cypherpunks mailing list (1994-09-09), "Thank you Bob Anderson", retrieved from http://web.archive.org/web/20080404222417/http://cypherpunks.venona.com/date/1994/09/msg00304.html

[7] R. Dube, Hardware-Based Computer Security Techniques to Defeat Hackers, From Biometrics to Quantum Cryptography, John Wiley and Sons, Inc., 2008.

[8] M. Stipčević, Quantum random number generators and their use in cryptography, Proceedings of 34nd International Convention MIPRO,2011.

[9] H.-Q. Ma et al., "A Random Number Generator Based on Quantum Entagled Photon Pairs", Chin. Phys. Lett., Vol. 21, No. 10, 2004

[10] P. L'Ecuyer, R. Simard, "TestU01: A C Library for Empirical Testing of Random Number Generators", ACM Transactions on Mathematical Software, Vol. 33, article 22, 2007.

[11] DIEHARD battery of stringent randomness tests, G. Marsaglia, available at http://www.stat.fsu.edu/pub/diehard/

[12] Dieharder: A Random Number Test Suite, R. G. Brown, available at http://www.phy.duke.edu/~rgb/General/dieharder.php

[13] A Statistical Test Suite for Random and Pseudorandom number generators for cryptographic applications, NIST Special Publications 800-22, group of autors, available at http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22b.pdf

[14] R. Fielding et al., "RFC 2616: Hypertext Transfer Protocol – HTTP/1.1", 1999.