Important comments regarding the 2006 white paper

Info at page 2

Firstly, it should be noted that the information on page 2 is no longer all up to date. The address of Alain H. Schumacher is now: Schulstr. 2-4, Weidingen, Germany. The telephone number is still current, but the fax number no longer exists. The email sicapas@pt.lu is also still up to date.

Concerning the registered patents: The patent for AHS-RNG was granted for cryptological applications and is still valid in the USA, Germany, Great Britain and France. However, they expire during 2026. For RPP-OTP, we have withdrawn the application, but we have achieved the main goal of ensuring that no one else can apply for patents on it by officially publishing the patent application.

New extension with 64-bit

AHS-RNG: Since the original version computes on a 32-bit arithmetic, we created a second version with 64-bit arithmetic to generate 2 bits with each computation. Therefore, the output on the odd digits is generated from the left half of the 64-bit, the even digits from the right half. Actually, two generators are now running in parallel, each with a different bit-fishing-table and different LCG parameters. The speed is now (on CPUs EPYC 2 Model 7702P) over 650 megabits per second. Since our small HPC has 1024 cores (16 servers of 64 cores each) we can generate 650 gigabits per second. This means that on our HPC we can generate a 256-bit secret crypto key for each of the nearly 8 billion people in less than 4 seconds, with better statistical quality than hardware based random number generators.

To create the BFT (Bit Fishing Table) we now use the AHS-RNG secret. However, for the 64-bit version, both BFTs have to be additionally tuned for equality/inequality per bit. In addition to the condition that the ratio of the 0-bit to the 1-bit must be 50/50, it is necessary that both parallel BFT tables are 25% each with 0-0,0-1,1-0 and 1-1. We found that otherwise a small statistical uncertainty becomes apparent, since in one out of 65536 cases the two bits happen to have the same address (at 8K BFT). But if, for example, 28% of the cases are 0-0, there is a small preponderance that no longer corresponds to the probability of perfect coin-tossing.

Tests AHS-RNG in 2006

The pages 13 to 18 (TEST RESULTS / AHS Random) are no longer current, the tests became more detailed and extended after 2006. Until 2006, the tests were conducted on three Pentium 3, while in the last 30 months 1024 (faster) cores were available on the HPC. In a separate article on the download page (4th line) you can find further explanations about the design of the current tests. In order to carry out all evaluations of the 300 Terabytes test-

results in the best possible way, we are still looking for the cooperation of motivated statisticians.

On page 17 of the 2006 white-paper we had written:

"We must apologize for not yet having calculated the probability of the different variants, as these probability values would increase the usefulness of this test. Did we hear someone say he will take over this challenge? "

Unfortunately, no one had the guts to do these calculations, so we did it ourselves. Perhaps we had neglected to offer a reward? The result of the combinatorial calculations can be found in text form as point 3 in the download area. If you are interested in an excel-file or a bc-file, please contact us via email.

Concerning the use of the LCG

The integration of the LCG is not a necessary part of the procedure, any similar PRNG can perform the same services, be it e.g. the MT19937 or the XOshiro256**. For the 8K version, we now use the front 16 bits of the generated 64-bit random number. This allows 2high48 (times 2 in the 64-bit version) bits to be generated before the back bit starts repeating.

Cryptography

It should be noted that the RPP-OTP method together with AHS-RNG is guaranteed to withstand all quantum computer attacks for the next 10,000 years. Since we did not receive any support in the implementation (only one European intelligence service expressed interest, but this did not lead to any results), and after we learned that in the USA a seller of cryptographical products was threatened with 17 years prison sentences because he had carelessly (?) sold his system to the wrong people, we decided not to pursue the RPP-OTP project any further.

AHS-RNG variants

Our current development is primarily aimed at applications in science. On the one hand, there is the possibility to easily generate correct random numbers with AHS-RNG in the deterministic version, which can be reproduced with the same BFT table and the same parameters for the LCG and the same FAAP table. For secret keys, there is then the non-deterministic version (AHS-RNG secret) which provides non-reproducible secret random numbers. If, however, you want non-deterministic true random numbers, but need to be able to reproduce them again, the optimal solution is the combination AHS-RNG record with AHS-RNG replay. In contrast to the physical RNGs, it is sufficient to store only the mini-entropy in a file. The size of this file is only 1/1700 of the volume of random numbers generated and still allows the generation of identical true random numbers. With a physical RNG you have to store all the random numbers in order to use them again. Our method allows to be used in cases where there is no sufficiently large and sufficiently fast storage

facility. For example, one terabyte is sufficient for 1,7 petabytes random numbers with the AHS-RNG record method, compared to the 1.7 petabyte requirement with physical RNGs.

Speed

Where there is an advantage, there is usually also a disadvantage: since every single bit is created in this method, the PRNG methods (e.g. MT19937 and XOshiro256**) are between 10 and 30 times faster. We had already written in 2006: "In the field of scientific simulations the tests have shown that it may enter the racecourse without wrong modesty; not in order to become the most powerful horse, but maybe to become the best horse for replacing by software the physical random generators in simulations". We think this statement is always valid.

However, one has to look at this in each individual case. What is the relationship between random numbers and the other calculations? Do all calculations take place in L1, L2 or L3 caches or does it need a high proportion of RAM memory? On our HPC, for example, we have had good experiences by splitting it between 2 kernels as master-slave. The slave is only responsible for generating the random numbers and sends them block by block via the L3 cache to the master (shared memory with semaphores).

All other comments, findings and the glossary are still up-to-date.

Author: Alain Schumacher / Weidingen, 7 August 2023