Presentation of the AHS-RNG sec-det (secret-determ) as the fifth member of the AHS-RNG family.

Today we would like to introduce a new variant in the AHS-RNG system. It is primarily a contribution to the philosophical clarification of the question of whether computers, with adequate programming and utilizing almost infinitely different parameters, can also generate deterministic "true" random numbers. This question arises for anyone who does not want to use PRNGs for their simulations, but instead strives for calculations with "true" random numbers.

The problem

If we want to perform a simulation with a physical RNG, we need a sufficient quantity of appropriate random numbers. If we generate these random numbers specifically for our experiment, we also have to store them if we want to repeat the experiment, for example with different parameter settings or an improved program. This has been known since 1949 and was already considered a major problem at that time. Just think of a simulation today on 100,000 CPU cores, each with 10^12 random numbers. Where do we get the TRN from, and where do we store it?

Our wish: new, fresh random numbers

Understandably, you don't want to use random numbers that have already been stored for another experiment; you want to use new, fresh random numbers. Although there are stored parameter files for AHS-RNG determ, they are predefined for the entire stream when they are created. As an example, we have a 16 GB file with 1,000,000 predefined parameters available for our tests. An alternative is, for example, the AHS-RNG record, which generates new, secret random numbers, but allows them to be replayed by reading the recorded "mini-entropy" with AHS-RNG replay. This only requires 1/1700 of the generated random numbers to be stored. Locally, this is already a big advantage, but if an institute on the other side of the world needs these files, it becomes difficult. From 3 times 50,000 Bigcrush tests, we have learned that the deterministically generated random numbers have the same quality as those produced with AHS-RNG secret. So the missing piece is the uniqueness and randomness of the required random numbers.

The solution: AHS-RNG sec-det

This is where the new AHS-RNG sec-det comes into play. It is, so to speak, a two-stage rocket. First, AHS-RNG secret generates enough secret random numbers to create a new, unprecedented BFT 8K (x2 in the 64-bit version), then a special FAAP and one or two seeds (256 bits) to start the XOshiro256starstar as the engine. Tests have shown that XOshiro is easy to integrate and, thanks to its period of 2^256 (as opposed to 2^48 for LCG), is guaranteed never to enter a repetition or period. The world will surely end before that happens!

The second stage then fires after the BFT, FAAP, and seed have been created, in the deterministic variant. It is now up to the user to decide whether to save these parameters, which are still secret, for later repetition. If the parameters are not saved, the random numbers generated remain secret forever, just as would be the case with fresh random

numbers from a physical TRNG. However, the AHS-RNG sec-det has the invaluable advantage that only 17 KB per stream needs to be saved.

Calculation of an example

If, for example, 100,000 cores ran with 10^12 random numbers, only 1.7 GB (for the 64-bit version) would now need to be transmitted to anyone who wanted to recalculate the experiment. This is no problem using special transfer services. With physical TRNGs, however, this would amount to 10^17 64-bit values, or 800 petabytes. With the AHS-RNG record and replay, it is "only" 471 terabytes.

Required storage space for 100,000 CPU cores with 10^12 64-bit random numbers each

Physical TRNG	800,000,000,000,000,000 bytes	= 800 petabytes
AHS-RNG record/replay	471,000,000,000,000 bytes	= 471 terabytes
AHS-RNG sec-det	1,700,000,000 bytes	= 1.7 gigabytes

But aren't all computer programs for random numbers PRNGs?

This is often claimed, but we would like to briefly explain why this does not apply to the AHS-RNG. The AHS-RNG is the only RNG to date that does not generate its random numbers using sophisticated mathematical formulas, but rather through combinatorial imitation of a coin toss from the continuous compilation of "0" or "1". How this works is clearly explained here at sicap.lu (navigation bar "AHS-RNG Demo"). Each "0" or "1" that contributes to the formation of a random number can come from one of the 65536 bit positions of the BFT.

The XOshiro256starstar has a unique period of 2^256 random numbers, after which it repeats itself. After one period, it has generated all possible random numbers equally often. Therefore, it is not a TRNG, but a PRNG. However, with XOshiro as its engine, the AHS-RNG does not have one period, but BFT (10`19725 possible) x FAAP (over 10^12 possible) and seeds (10^77), which together result in 10^19814 possible start values. Each start value generates an independent random number stream of over 10^75 random numbers. These random numbers will **not** be evenly distributed, but will follow the laws of probability and achieve a normal distribution. Only the probability of generating all possible values is evenly distributed!

How big can you imagine 10^19814 to be?

To say it up front: No human, no expert, and no AI can even begin to imagine how large this number is. It is beyond the human imagination, and also that of AI! That is why we speak of "almost infinite." We dare to claim that no physical TRNG can draw on such a number of quantum states to justify its randomness.

When will program versions of the AHS-RNG family be available for download at sicap.lu for testing?

Since we have been gradually trying out different variations for 18 years, we now need to create an official version of each variant. Then we need to perform a final check of each variant with 10,000 bigcrush runs. Realistically, you can expect to find the officially published program versions under the Christmas tree!