Heads and tails, summer 2015

PROF. DR. JOACHIM VON ZUR GATHEN AND DR. DANIEL LOEBENBERGER

6. Exercise sheet
Hand in solutions until Sunday, 31 May 2015, 23:59:59

This exercise sheet is conceptually a little bit different from typical ones. It asks you to study the generators presented in the lecture. The goal is to have at the end a table of comparable results for all of the generators. For solving the task all of you have to work as one research team. First decide in which language you want to work in (we strongly recommend plain old C with some large integer arithmetic library such as `libgmp`) and which libraries to use. You can use our mailing list for discussion. Distribute subtasks to the different members of your team.

Exercise 6.1 (Analyzing the generators discussed). (20+20 points)

On the course webpage you find a 500kB binary file which are the output of a concrete run of the hardware random generator presented in the lecture. 20 +20

(i) Compute the distributions of bytes, i.e. count for every byte how often it occurs in the file. Hint: You have done this already, just recall your former results. The same applies for the next question.

(ii) Compute the byte entropy of the distribution. What is the maximal value?

Use the bits in the file provided sequentially to produce any random data that you need. In each case, state how many bits you use. For each of the six generators in (iii) through (viii), generate 500kB of output, measure the time that this takes, and compute the byte entropy of your output. State which hardware and software environments you use, and how you measure time. Before measurement, restart your machine and make sure no other processes run concurrently. Prepare a table with byte entropies, runtimes and two further columns. One of them contains the throughput in kB/sec, and the other one a normalized form of this. Namely, you take t_0 to be the smallest throughput of all your generators and present for each other one its throughput divided by t_0.

(iii) Analyze `/dev/random` or a similar source of random bytes on your machine.

(iv) Analyze the linear congruential generator with $m = 2^{1279} - 1$ prime. Generate s, t and x_0 from the source of random bytes provided.

(v) Analyze the Littlewood generator with $n = 5$ and $d = 7$ in its decimal version.

(vi) Analyze the RSA generator for a randomly chosen 3000 bit RSA modulus N (as explained below), a randomly selected public exponent $e \in \mathbb{Z}_N^\times$, and a randomly generated seed, outputting ten bits per round. Hint: To compute the modulus, you generate a 1500 bit number a. Then you compute deterministically the next larger number $b \geq a$, such that $b = 3$ in $\mathbb{Z}_4$ and $\gcd(b, m) = 1$, where m is the product of all primes up to 1000. Once you found b, you test whether it is prime and return $p = b$ if it is or repeat the whole process if not. State the values of a and b that you used.

(vii) Analyze the Nisan-Wigderson generator for the (k, n, s, t)-design D from Exercise 5.2 (iv) (you also find a description in the lecture notes, chapter 1.7) with $k = 121$, $n = 1331$, $s = 11$ and $t = 2$. Use the function $f_D \colon \mathbb{B}^k \to \mathbb{B}^n$ for the following function $f \colon \mathbb{B}^s \to \mathbb{B}$ as the building block: On input $(x_0, \ldots, x_{10})$, the function f computes the parity of the sum of the numbers $(x_0, \ldots, x_5)$ and $(x_5, \ldots, x_{10})$ interpreted as elements of $\mathbb{Z}_{2^6}$. Use a random seed $x \in \mathbb{B}^k$ and the construction from the lecture to produce long generators from short ones for the generator.

(viii) Analyze the Blum-Blum-Shub generator with the same parameters as in subtask (vi).

(ix) Perform further experiments and interpret your results.