The art of cryptography: security, reductions, and group cryptography

Prof. Dr. Joachim von zur Gathen, Konstantin Ziegler

9. Assignment

(Due: Thursday, 14 June 2012, 13^{00})

Exercise 9.1 (generating (safe) primes)**.** The expected runtime of an algorithm that generates primes of a special form usually depends on the number of such primes up to a given bound. In this exercise, we consider primes of the following form.

$$\begin{aligned} P(x) &= \{\text{all primes} \leq x\}, \\ P_1(x) &= \{p \in P(x) \colon p = 2p_0 + 1 \text{ for some prime } p_0\}, \\ P_2(x) &= \{p \in P(x) \colon p = 2p_0q_0 + 1 \text{ for some primes } p_0, q_0\}, \end{aligned}$$

and denote their sizes by $\pi(x)$, $\pi_1(x)$, and $\pi_2(x)$, respectively.

Assuming the Riemann hypothesis, we have the following approximation for the density of the first set.

$$\frac{\pi(x)}{x} = \frac{\mathrm{Li}(x)}{x} + O\left(\frac{\log x}{\sqrt{x}}\right), \tag{9.2}$$

where $\mathrm{Li}(x) = \int_2^x \log t \, dt$ is the *logarithmic integral*.

[For the following tasks, use subroutines or libraries that provide fast primality testing and evaluation of Li. These are widely available. Do not try to implement them yourself!]

(i) (4 points) Consider the following naive algorithm to generate primes up to size x.

Algorithm 1: Prime(x)

Input: integer x
Output: prime $p \leq x$

```
p ← 1
while p is not prime do
    p ←R [1, ..., x] uniformly
end
return p
```

Let $x = 100$ and run the algorithm H times (choose H reasonably depending on the speed of your primality test) and count the total number N of loop executions. Use the quotient H/N as approximation for $\pi(x)/x$ and derive a value for the implicit constant C in (9.2) from that.

Repeat this experiment for increasing values of x and plot $C(x)$.

(ii) (3+1 points) Let us now generate Sophie-Germain primes.

Algorithm 2: $\mathsf{Prime}_1(x)$

```
Input: integer x
Output: prime p ∈ P_1(x)
p ← 1
while p is not prime do
    p_0 ← Prime(x/2 − 1)
    p ← 2p_0 + 1
end
return p
```

Again, use the average number of loop executions as approximation to $\pi_{SG}(x)/(x/2)$, the fraction of Sophie-Germain primes among all odd numbers $\leq x$. Let us assume that this will be

$$\frac{\pi_1(x)}{x/2} = \frac{2\,\mathrm{Li}(x)}{x} + O\left(\frac{\log x}{\sqrt{x}}\right).$$

Run this for increasing values of x and plot the development of the implicit constant. Comment on your observation.

(iii) (3+2 points) Finally, let us generate primes of the form $2p_0q_0+1$ for primes p_0, q_0 with the following algorithm.

Algorithm 3: $\mathsf{Prime}_2(x)$

```
Input: integer x
Output: prime p ∈ P_2(x)
p ← 1
while p is not prime do
    p_0 ← Prime(x/2)
    q_0 ← 1
    while q_0 ≤ p_0 do
        q_0 ← Prime(x/2p_0 − 1)
    end
    p ← 2p_0q_0 + 1
end
return p
```

Let us conjecture that the average number of loop executions will be

$$\frac{\pi_2(x)}{x/2} = \frac{2\,\mathrm{Li}(x)}{x} + O\left(\frac{\log x}{\sqrt{x}}\right).$$

Do your experiments support or refute this conjecture? In the latter case, can you come up with a conjecture supported by your experiments?