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

Prof. Dr. Joachim von zur Gathen, Konstantin Ziegler

1. Assignment

(Due: Thursday, 12 April 2012, 13^{00})

Exercise 1.1 (Expected value of a random variable)**.** We are given a discrete random variable X, for example the result of a single roll of a fair die. The values that X can take are denoted by x and the respective probability is given by $\text{Prob}(x \xleftarrow{\$} X)$. For the example, the x are taken from the set $A = \{1, 2, 3, 4, 5, 6\}$ each with $\text{Prob}(x \xleftarrow{\$} X) = 1/6$.

We are interested in the *expected value* $E(X)$ defined as

$$E(X) = \sum_x x \cdot \text{Prob}(x \xleftarrow{\$} X),$$

where the sum is taken over all possible values of X. In the example above, this returns as the expected value for the roll of a single fair die

$$E(X) = \sum_{x \in A} x \cdot \frac{1}{6} = \frac{21}{6} = 3.5.$$

Next, we roll the die until a certain number, say "2", appears *for the first time*. The random variable Y is now the *number of rolls* that are performed, until this happens.

(i) (2 points) What is $\text{Prob}(y \xleftarrow{\$} Y)$, i.e. the probability that "2" appears for the first time in the yth roll?

(ii) (4 points) Prove $E(Y) = 6$. (Hint: $\sum_{k=n}^{\infty} q^k = q^n/(1-q)$ for $|q| < 1$.)

(iii) (3 points) Generalize the preceding steps to prove the more general statement.

> Suppose that an event A occurs in an experiment with probability p, and we repeat the experiment until A occurs. Then the expected number of executions until A happens is $1/p$.

Exercise 1.2 (Empirical security of RSA)**.** For the security of RSA, one has to consider the difficulty of factoring large numbers (which are a product of two primes). The general number field sieve achieves a (heuristic, expected) running time of $2^{((64/9+o(1))n\log_2^2 n)^{1/3}}$ for n-bit integers. It is not correct to think of $o(1)$ as zero, but for the following rough estimates just do it. Factoring the 663-bit integer RSA-200 needed about 165 1GHz CPU years, that is 165 years on a single 1GHz Opteron CPU, using the general number field sieve.

(i) (3 points) Estimate the time that would be needed to factor an n-bit RSA number assuming the above estimate is accurate with $o(1) = 0$ (which is wrong in practice!) for $n = 1024, 2048, 3072$.

Exercise 1.3 (RSA key generation on light-weight devices)**.** We investigate the effect of seeds "with little randomness" on two procedures to generate RSA moduli. The pseudocode for the the first procedure reads as follows.

```
1. pseudo-random_string = expand(seed)
2. p = generate_random_prime(pseudo-random_string)
3. q = generate_random_prime(pseudo-random_string)
4. return N = p*q
```

All procedures are deterministic and the only (possible) variation in each execution is in the seed to the function expanding it to a pseudo-random string.

(i) (3 points) Assume a seed with 20 bits of "true randomness". After how many calls do you expect a value for N that was already generated before? After how many calls a value for N that has a nontrivial gcd with one of the previously generated values?

Some implementations, for example OpenSSL's RSA key generation, add some randomness after generating the first prime, with the intention of increasing security.

```
1. pseudo-random_string = expand(seed)
2. p = generate_random_prime(pseudo-random_string)
3. pseudo-random_string += add_randomness(bits)
4. q = generate_random_prime(pseudo-random_string)
5. return N = p*q
```

(ii) (4 points) Assume a seed with 20 bits of "true randomness" and also 20 bits for the added randomness. Answer the questions of (i) for this second procedure.