5. Exercise sheet
Hand in solutions until Saturday, 30 November 2013, 23:59:59

Exercise 5.1 (Reductions for RSA). (7+1030 points)

We consider as an attacker a (probabilistic) polynomial-time computer $A$. $A$ knows $pk = (N, e)$ and $y = enc_{pk}(x)$. There are several notions of “breaking RSA”. $A$ might be able to compute from its knowledge one of the following data.

- $B_1$: the plaintext $x$,
- $B_2$: the hidden part $d$ of the secret key $sk = (N, d)$,
- $B_3$: the value $\varphi(N)$ of Euler’s totient function,
- $B_4$: a factor $p$ (and $q$) of $N$.

If $A$ and $B$ are two computational problems (given by an input/output specification), then a random polynomial-time reduction from $A$ to $B$ is a random polynomial-time algorithm for $A$ which is allowed to make calls to an (unspecified) subroutine for $B$. The cost of such a call is the combined input plus output length in the call. If such a reduction exists, we write

$$A \leq_p B.$$

(i) Show that $B_1 \leq_p B_2$. 2 points
(ii) Show that $B_2 \leq_p B_3$. 2 points
(iii) Show that $B_3 \leq_p B_4$. 2 points
(iv) Which problem is the easiest one? Which one is most difficult? 1 point
(v) Show that additionally we have $B_4 \leq_p B_3$. Hint: Consider the quadratic polynomial $(x - p)(x - q) \in \mathbb{Z}[x]$. 2 points
(vi) Argue that we also have $B_3 \leq_p B_2$. 4 points
(vii) Resolve the question whether also $B_2 \leq_p B_1$ or equivalently whether $B_4 \leq_p B_1$. Warning: This is an open research problem... 1024 points
Exercise 5.2 (RSA bad choice). (6 points)

Show why the 35-bit integer $23360947609$ is a particularly bad choice for $N = pq$.

We claim that two prime numbers which are really close to each other are bad choices for RSA system. To show this we use Fermat’s factorization method based on the fact: If $N = pq$ with $p > q$ being odd primes, then $N = \left(\frac{p+q}{2}\right)^2 - \left(\frac{p-q}{2}\right)^2$.

(i) Explain how you can use this fact to find prime factors of $N$.

(ii) Do it for $N = 23360947609$.

Exercise 5.3 (Primality Testing). (10+10 points)

In this exercise we put hands on the primality tests discussed in the lecture.

(i) Implement the Fermat test in a programming language of your choice.

(ii) Implement the Strong pseudoprimality test in a programming language of your choice.

Now, let’s run it! Execute the Strong pseudoprimality test with

(iii) $N = 41$, $x = 2$.

(iv) $N = 57$, $x = 37$.

(v) $N = 1105$, $x = 47$.

(vi) $N = 1105$, $x = 2$.

With our implementation running, we can now perform several experiments.

(vii) Compute the number of Fermat liars for $N = 35$, i.e. the number of choices $x \in \mathbb{Z}_N$ for which the Fermat test returns “$N$ is possibly prime”.

(viii) Compute the number of Strong liars for $N = 35$, i.e. the number of choices $x \in \mathbb{Z}_N$ for which the Strong primality test returns “$N$ is probably prime”.

(ix) Do the same for $N = 561$.

(x) Perform more experiments.

(xi) Interpret the results.