Practical Exercises 2 - RSA

September, 28^{th}, 2020

The goal of these exercises is to familiarize with RSA's main weaknesses. For the implementation, you will use `sage`, a python-based mathematical toolbox.
A report is expected, as a tar file containing your code as long as your answers to the questions. Do not forget to comment your code, and give details in your answers. Include **any leads you explored**, even if they did not succeed, and **as much details as** you can to show your understanding of the subject.

You shall send your reports in an e-mail, containing "[BCS]" in the subject, to `daniel.de-almeida-braga@irisa.fr`. The report is expected within a week after the session.

Exercise 1: Factorisation

When prime numbers have some specific properties, we can use some specific factorization algorithm taking advantage of this property to make factorisation easier.

> You can find the file `tp2-rsa_material.sage` on the share (accessible using your student logins), containing all necessary data.

1. Generating large prime numbers can take some time, so Alice decided to generate only one. Can you find out a way to factor `n1` and recover the message ?
   Hint: finding the square root is easier than factoring!
   (a) Given the factor, you can compute $\varphi(n_2)$, but be careful and take a look at the formula[1].
   (b) Using $\varphi(n_2)$, you can easily recover the full private key (`inverse_mod` should be helpful here).
   (c) Recover the message.
2. Implement the Pollard p-1 algorithm to factorize an integer.
3. Using this algorithm, try to factor `n2` and decrypt the message.

Exercise 2: Decryption oracle

Given a ciphertext `c`, a public key `(n, e)` and an oracle allowing to decrypt any message (except `c` !), how can you recover the plaintext corresponding to `c` ?
Hint: Remember that $(a * b)^e \bmod n = a^e * b^e \bmod n$

Once you get the idea, you can try it by solving "RSA - Decipher Oracle" on root-me for 25 points.

[1] `https://en.wikipedia.org/wiki/Euler's_totient_function#Value_for_a_prime_power_argument`

[1] `https://en.wikipedia.org/wiki/Pollard's_p_-_1_algorithm`