Practical Exercises 2 - RSA

September, 09^{th}, 2021

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.

You can contact me anytime on discord (Cyberschool server) or by mail at `daniel.de-almeida-braga@irisa.fr`.

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 drive, containing all necessary data.

1. Generating large prime numbers can take some time, so Alice decided to generate only one (and use $p = q$). 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 $(a \times b)^e \bmod n = a^e \times b^e \bmod n$

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`