**[Politechnika Krakowska im. Tadeusza Kościuszki](http://www.pk.edu.pl/)**

**1st Semester – 2015/2016**

**Cryptography**

Pedro Dias Faria – [pedrodiasfaria@gmail.com](mailto:pedrodiasfaria@gmail.com)

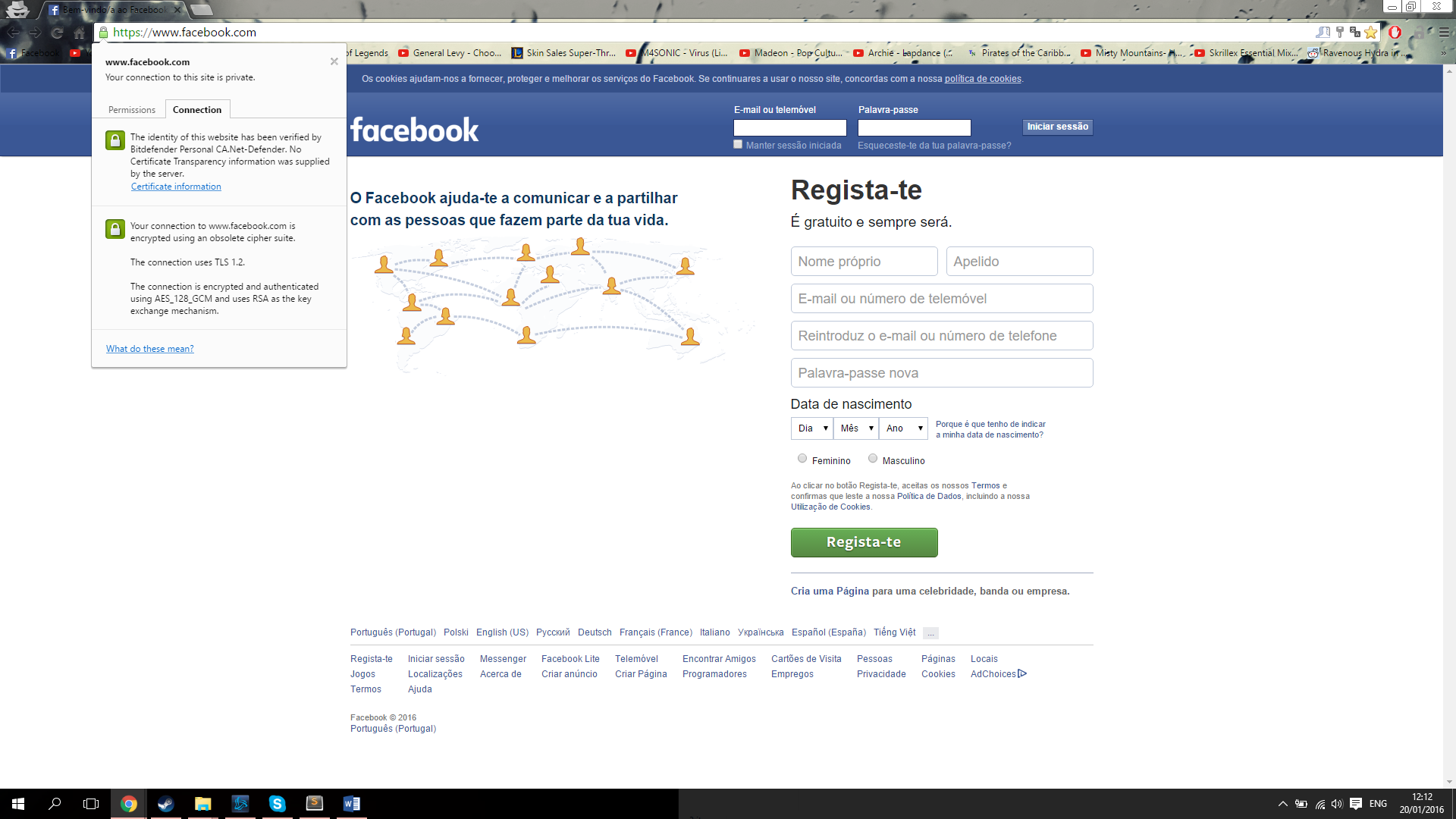
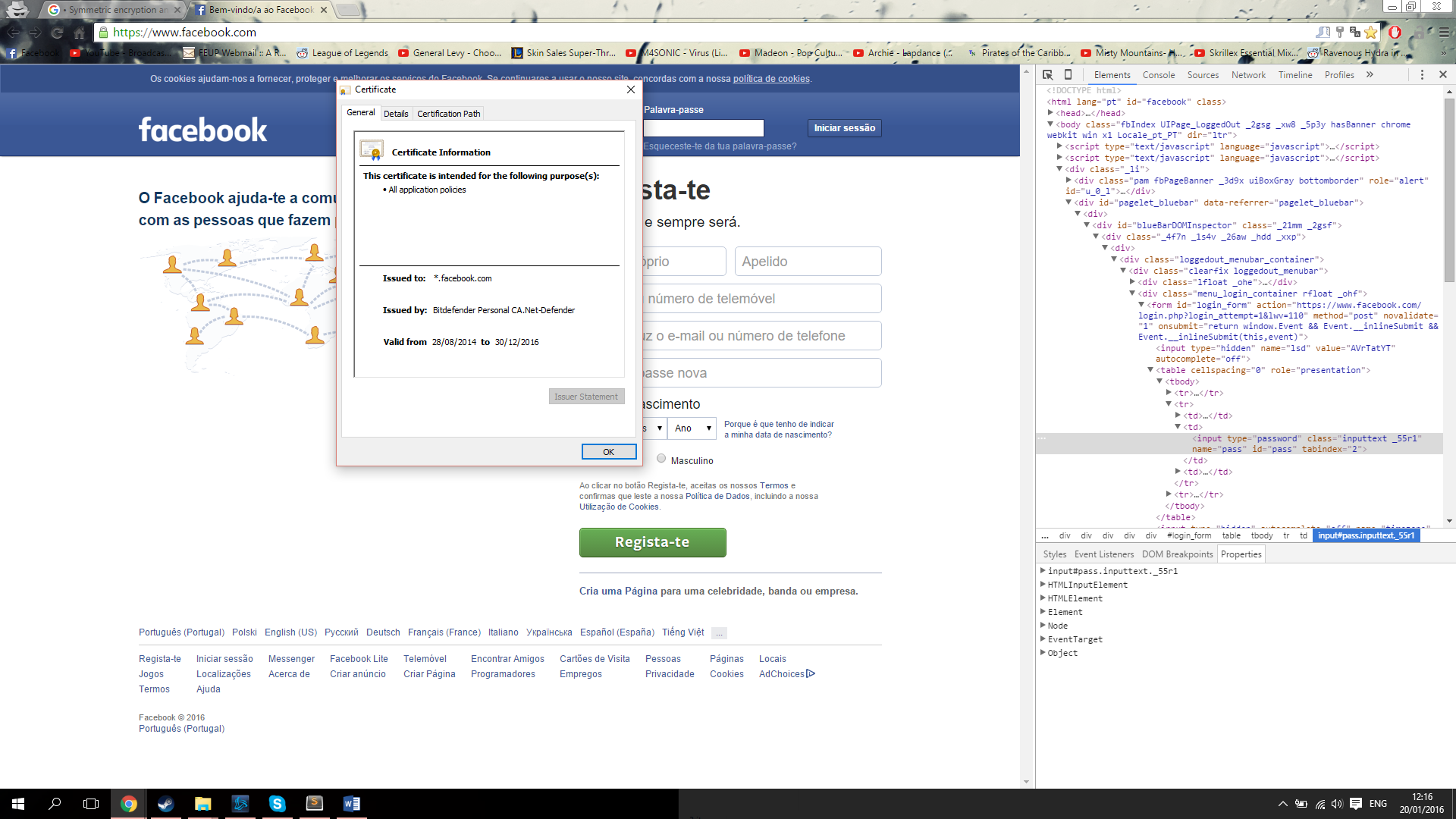
Description

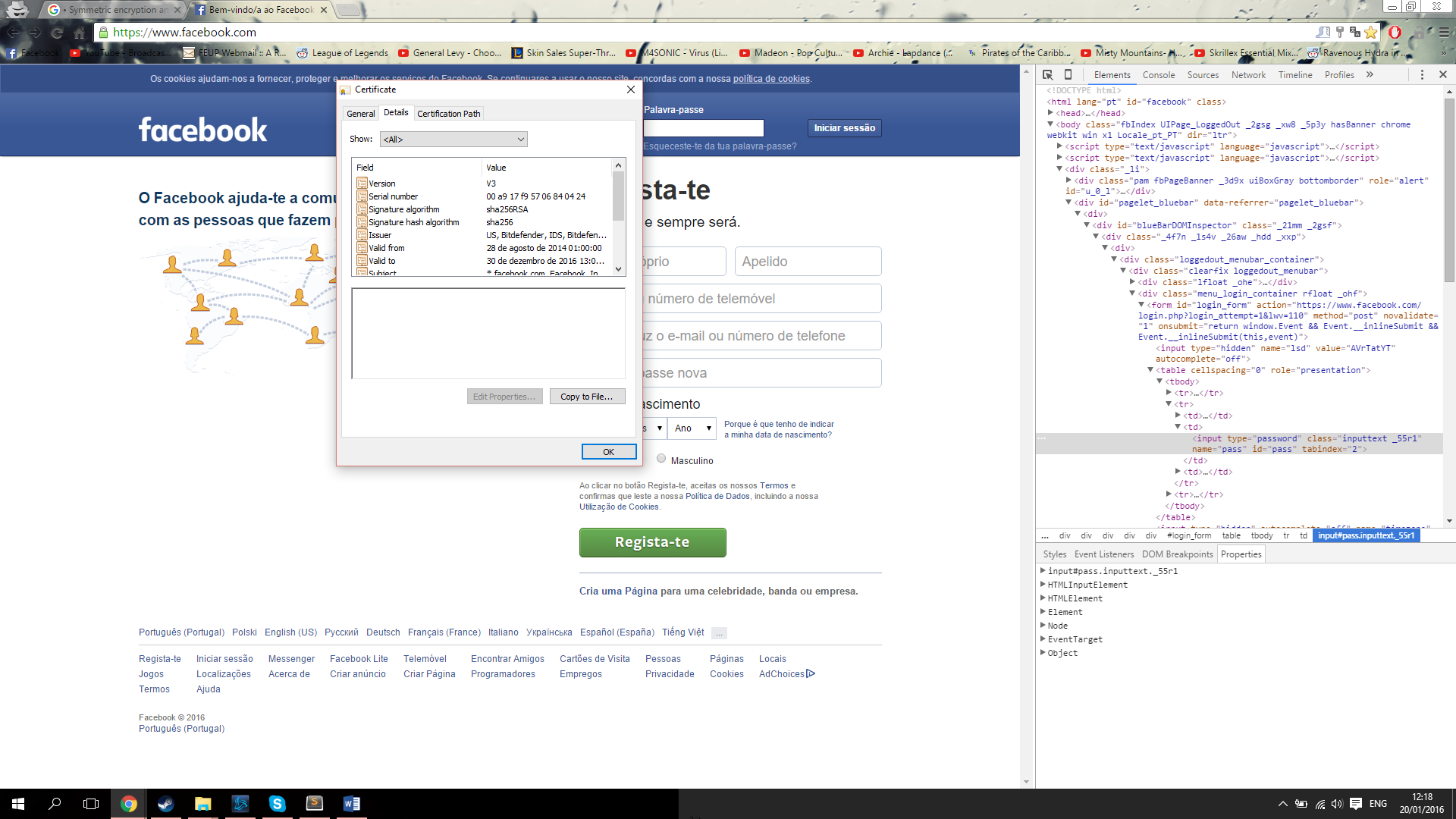
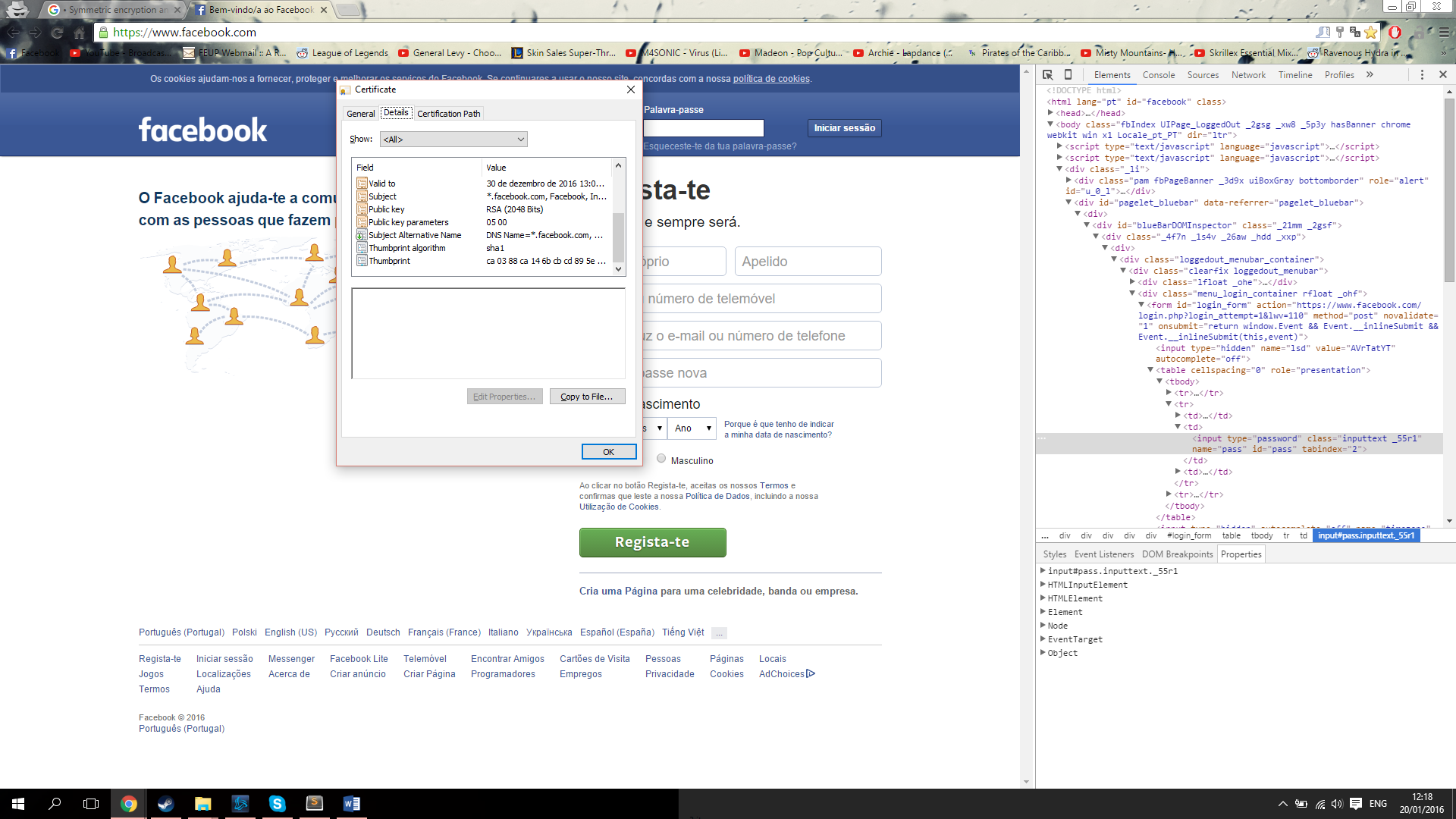
This is a little compilation of the objectives of each project and how to run the programs.

Project 1 – Cryptographic Protocols

In the application we are using (chosen facebook)

From the <https://www.facebook.com> :

Project 2 – Encrypting Letters with RSA

Using the first letter of my Name and Surname, being A equivalent to 0:

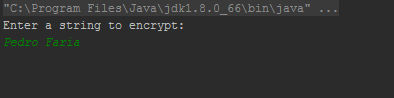
P = 15 and F = 5

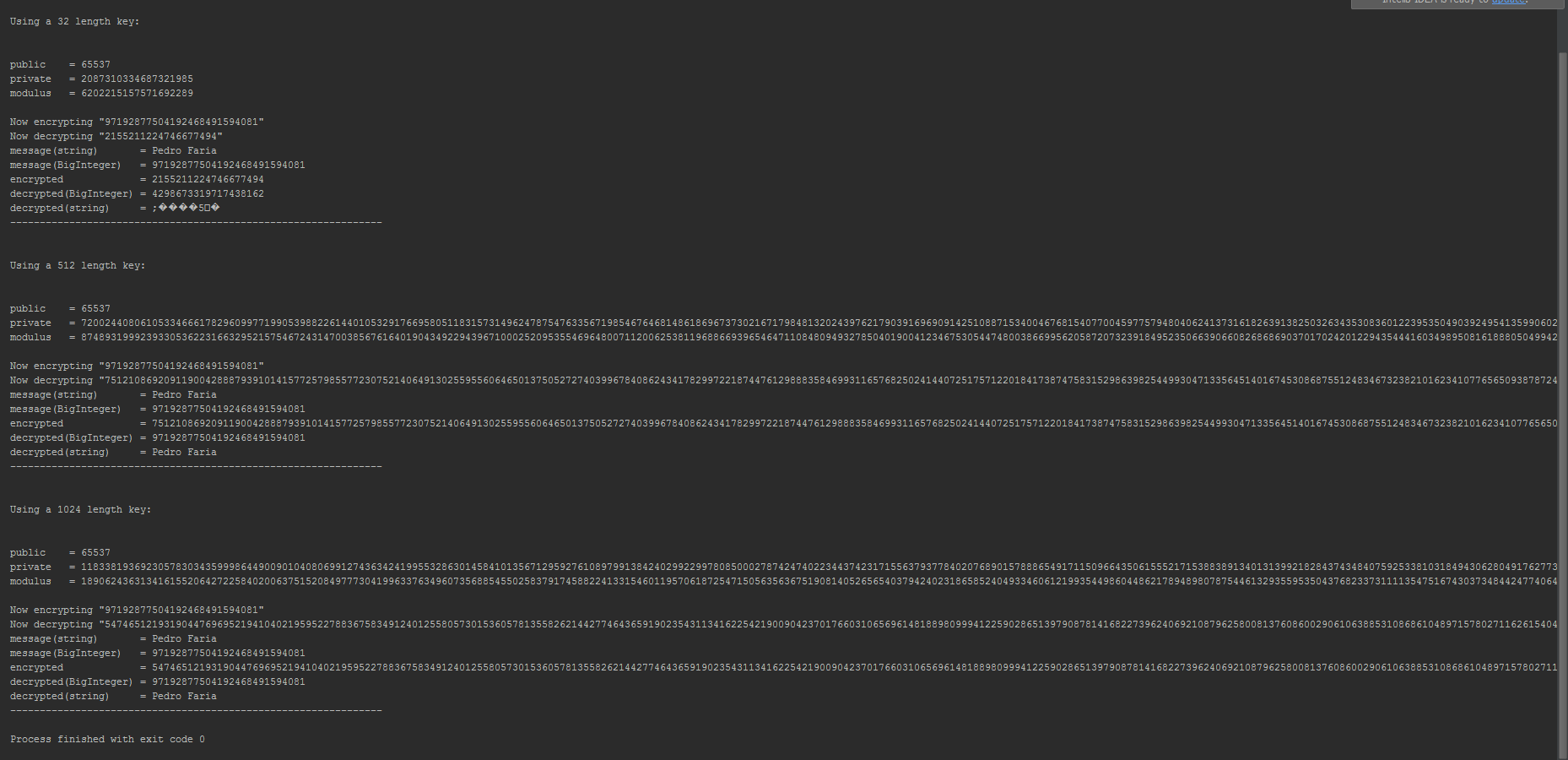
1. With RSA, initially the person picks two prime numbers. For example:
   1. p = 11 and q = 3
2. Then n is calculated as n = p x q:
   1. n = 11 x 3 = 33
3. Then phi being phi = (p-1)(q-1)
   1. phi = 10 x 2 = 20
   2. Being the factors of phi:  1, 2, 4, 5, 10 and 20
4. Next the public exponent e is generated so the gcd of e and phi is 1
   1. The smallest value: e = 3
5. The factors of e are 1 and 3, being 1 is the highest common factor of them. n and e values are the public keys. The private key d is the inverse of e modulo phi: d=*e*^(-1) mod [(*p*-1)x(*q*-1)]
   1. d = 3^(-1) mod 20 = 7
   2. So now we know Public Key (n, e) = (33, 3)
   3. And Private Key (n, d) = (33, 7)
6. With all keys known, we can encrypt/decrypt the message:
   1. Being m the message, c the encrypted message:
      1. c = m^(e) mod n =
         1. for P: c = 15^3 mod 33 = 3375 mod 33 = 9
         2. for F: c = 5^3 mod 33 = 125 mod 33 = 26
   2. And now to decrypt:
      1. m = c^(d) mod n
         1. for c = 9: m = 9^7 mod 33 = 4782969 mod 33 = 15
         2. for c = 26: m = 26^7 mod 33 = 8031810176 mod 33 = 5

Project 3 – Simple RSA

To run, execute main.java

It will ask for a input string, and will calculate the RSA encryption/decryption of the input with random keys with 32, 521 and 1024 length each.





Project 4 – RSA cloud version

This program is the most complex of the projects.

Breaking in parts, it does the following:

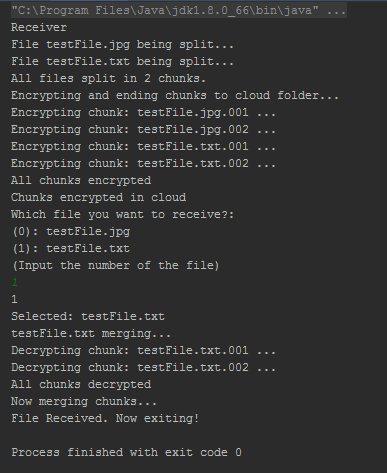
/\*  
 \* Steps:  
 \* 1. Select number of senders  
 \* 2. Divide files between each sender  
 \* 3. Each sender sends his chunk to the cloud encrypted  
 \* 4. Cloud keeps chunks encrypted  
 \* 5. Receiver asks for a file  
 \* 6. Cloud sends chunks to receiver, who then decrypts and merge  
 \*/

The most complex part is the encryption / decryption of files. Since we are using RSA cipher provided by java, we face 2 problems:

1. Ciphers use byte arrays, so every file has to be converted to one;
2. Block ciphers can’t encrypt arbitrary long byte arrays directly.

So depending on the mode (De- or Encryption), it will work with different sized byte arrays.

The execution of the program is easy. First put the files to test on the “FilesAvailabe” folder. Then after choose the file to retrieve, and it will appear on the “ReceiverFolder”.



Project 5 – MD5

To run, execute MD5.java

It must exist a file named “testFile.jpg” on the path ”./src/MD5/testFiles/testFile.jpg”

The program executes automatically an encryption of several strings, the file, and also 2 different strings with collision hashes.

All strings/files codifications can be confirmed on: <http://onlinemd5.com>

