Public Key Encryption: Part 1

IMPORTANT NOTES:

1. **Study lecture materials at least 1 hour and prepare Question 1-2 under Lab Tasks section prior to the lab session. Prepared questions will be discussed in the lab session.**

1 Overview

The objectives of this lab are to learn concepts of number theory on how to compute modular arithmetic, discrete log and multiplicative inverse in Sage.

2 Lab Environment

SageMath: In this lab, we need to use the SageMath library to perform certain mathematical calculations. Use the SageMath web interface at `https://sagecell.sagemath.org/`.

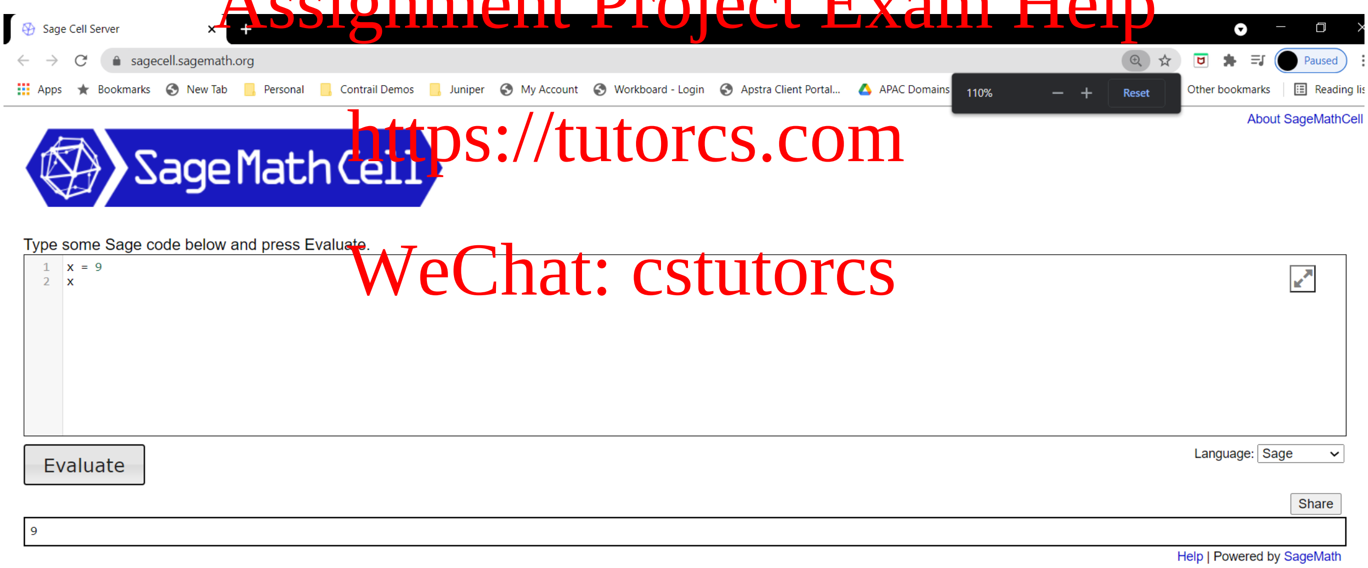

Figure 1: Sage Interface

3 Lab Tasks: Modular Arithmetic

All the numbers in this section are represented in base 10.

For the following task you can use the sage library variables similar to programming languages such as python. For instance writing x=7 will create an integer variable x with value 7 that can be used in other expressions. To print the value of a variable simply type its name and press enter. Note that mod can be performed with % in sage as illustrated in the last example below that computes $7^3 \bmod 11 = 2$. Examples:

- The sage code

  ```
  x=7
  y=9
  x
  ```

  returns (after pressing Evaluate):

  ```
  7
  ```

- The sage code

  ```
  x=7
  y=9
  x, y, x*y
  ```

  returns (after pressing Evaluate):

  ```
  (7, 9, 63)
  ```

- The sage code


  ```
  x=7
  y=9
  x**3
  ```


  returns (after pressing Evaluate):

  ```
  343
  ```


- The sage code

  ```
  x=7
  y=9
  x**3 % 11
  ```

  returns (after pressing Evaluate):

  ```
  2
  ```

1. For $a = 1305051748$, $b = 3528645214$, $p = 146347$, in the first step calculate $n_1 = a \times b$ and then $n_1 \bmod p$, for the second step calculate $r_1 = a \bmod p$, $r_2 = b \bmod p$ and then $r_1 \times r_2 \bmod p$

   ```
   a=1305051748
   b=3528645214
   p=146347
   n1=a*b
   r1=a % p
   r2=b % p
   n1, n1 % p, r1*r2 % p
   ```

returns the result:

```
(4605064604602534072, 75437, 75437)
```

2. For $a = 5$, $b = 22401562154533299041783378656$, $p = 43902608862042298666481977063$, calculate $a^b \bmod p$ by using `power_mod(a,b,p)`.

```
power_mod(5,22401562154533299041783378656,43902608862042298666481977063)

returns result

42568949792454651564941152870
```

3. For $a = 5$, $b = 42568949792454651564941152870$, $p = 43902608862042298666481977063$, solve the discrete logarithm $a^x \bmod p = b$ by using `discrete_log(b,Mod(a,p))`. Do you notice the time difference between solving the discrete logarithm and computing the multiplication modulo reductions in 3.1.2?

```
discrete_log(42568949792454651564941152870,Mod(5,43902608862042298666481977063))

returns result

22401562154533299041783378656
```


Solving the discrete logarithm usually takes longer time.

4. For $e = 65537$, $p = 43902608862042298666481977063$, solve the multiplicative inverse of $e \bmod p$ by using `inverse_mod(e,p)` and verify the answer by using `mod()`


```
e=65537
p=43902608862042298666481977063
d=inverse_mod(e,p)
d

returns result

1260063891687162424152045392


To verify it:
e=65537
p=43902608862042298666481977063
d=inverse_mod(e,p)
a=mod(e*d, p)
a

returns result

1
```

4 Optional Extra Exercises

(a) The following sage code generates two random 512-bit primes p, q, multiplies them to get $n = p \times q$ and computes $r = \gcd(n, p)$. What do you think should be the value of r in terms of p and q? Try to run the code and find out.

```
p=random_prime(2^511,2^512-1)
q=random_prime(2^511,2^512-1)
n=p*q
r=gcd(n,p)
p,q,r
```

Since p divides n, p is the GCD of n, so $r = p$.

(b) Experiment to see how the running time of sage code to compute a modular exponentiation (Given a, b, p, compute $y = a^b \bmod p$ for a prime p) depends on the bit-length of the numbers a, b, p by doing the computation with numbers of increasing bit length and making a graph of run-time versus bit length. Do the same experiment for the discrete-logarithm computation (given y, a, p, find b such that $y = a^b \bmod p$). Do you think your results match the expected behaviour (polynomial time for modular exponentiation, subexponential time for discrete logarithm)?