# Homework Assignment 02

Machiry Aravind Kumar

UCSB

## 1 Implement MonPro that returns the Montgomery product c and also a bool value StepFour which is True if there is a subtraction in Step 4 or False if not. Using the MonPro routine, create an implementation of the modular exponentiation algorithm MonExp

I have implemented MonPro and MonExp using Python. The code for which is present in the file `machiry_monpro_timing_attack.py`. Functions for MonPro and MonExp are `mon_pro` and `mon_exp_attacker` respectively.

## 2 Timing Attack

For a given $d$ and $n$, First I generate 10000 random messages. Second, I run *mon_exp_compute_total_subtractions* for each message $m$ to compute $m^d \pmod{n}$ along with total number of subtractions happened in all the `mon_pro` invocations. In code, variable `nums_subs` holds this.
Except for Most significant bit ($d_k$ which can be reasonably assumed to be 1), for each of the other bits of $d$, I assume $d_i$ $(i < k)$ to be 1 and group messages depending on return value of `mon_pro` i.e `true` (Group **A**) or `false` (Group **B**). In code, variable `d_arr` holds this.
For each bit $d_i$, compute the average number of subtractions of messages in Group **A** and Group **B**, if the difference is greater than 1, then predict $d_i$ to be 1 else predict 0.

## 3 Code usage and results

Attached is the file `machiry_monpro_timing_attack.py` that contains all the code. You can run the code as shown in the examples below:

### 3.1 Example 1 (d=85, n=391)

```
>> python machiry_monpro_timing_attack.py 85 391

Computing MonPro along with timings for: d=85, n=391 with no of messages = 10000
Computing Differences for each bit of exponent
[+] Timing Attack Results:
    For bit:1 difference:1.49956114749 Guessing bit to be 1
    For bit:2 difference:0.733689867916 Guessing bit to be 0
```

```
    For bit:3 difference:1.2635785391 Guessing bit to be 1
    For bit:4 difference:0.560659944641 Guessing bit to be 0
    For bit:5 difference:1.12327680701 Guessing bit to be 1
    For bit:6 difference:0.643323021801 Guessing bit to be 0
    For bit(MSB):7 difference: 1.9002 Guessing bit to be 1

[+] Attack Predicted : 1010101
[+] Actual Value : 1010101

++++++ VALUES MATCH: TIMING ATTACK SUCCESSFULL ++++++
```

### 3.2 Example 2 (d=6597, n=11413)

```
>> python machiry_monpro_timing_attack.py 6597 11413

Computing MonPro along with timings for: d=6597, n=11413 with
    no of messages = 10000
Computing Differences for each bit of exponent
[+] Timing Attack Results:
    For bit:1 difference:1.24845233437 Guessing bit to be 1
    For bit:2 difference:0.595160580067 Guessing bit to be 0
    For bit:3 difference:1.0800554905 Guessing bit to be 1
    For bit:4 difference:0.628992367776 Guessing bit to be 0
    For bit:5 difference:0.559395113874 Guessing bit to be 0
    For bit:6 difference:0.492662803506 Guessing bit to be 0
    For bit:7 difference:1.04541593116 Guessing bit to be 1
    For bit:8 difference:1.18725120832 Guessing bit to be 1
    For bit:9 difference:1.12435168822 Guessing bit to be 1
    For bit:10 difference:0.511415480353 Guessing bit to be 0
    For bit:11 difference:0.557207429916 Guessing bit to be 0
    For bit:12 difference:1.03862332563 Guessing bit to be 1
    For bit(MSB):13 difference: 3.7471 Guessing bit to be 1

[+] Attack Predicted : 1100111000101
[+] Actual Value : 1100111000101

++++++ VALUES MATCH: TIMING ATTACK SUCCESSFULL ++++++
```