CSE451 Computer & Network Security

Assignment 2: Toy RSA Algorithm using Montgomery Multiplication

Deadline: 2025-12-24 Wednesday 11:59pm

It is required to implement a toy RSA algorithm using Montgomery multiplication in C. The modulus N should fit in a 32-bit integer. The I/O data is parsed and written as hexadecimal strings.

$$N = p * q$$

$$\phi(N) = (p-1)(q-1)$$

$$d = e^{-1} mod\ \phi(N)$$

$$C = P^e\ mod\ N$$

$$P = C^d\ mod\ N$$

Required build command:

```
    gcc  -O3  studentID.c  -o  studentID
```

The submission is one C source file. No need to use any external libraries or headers.

Required program usage:

```
    studentID.EXE  "g"  public_key.txt  private_key.txt
    studentID.EXE  "e"  public_key.txt  plaintext.txt  ciphertext.txt
    studentID.EXE  "d"  private_key.txt  ciphertext.txt  plaintext.txt
```

The command arguments (keys, plaintext, and ciphertext) are all filenames. Use fopen(), fread(), fwrite(), and fclose() to access the data.

All files are written in hexadecimal.

Example file contents:

File name	File contents	Description
public_key.txt	00000101 0001E709	(e, n)
private_key.txt	0000CF01 0001E709	(d, n)
plaintext	00004567	data block < n
ciphertext	0001E5CF	data block < n

Montgomery Multiplication

The naive modular reduction ('%' in C) involves a DIV instruction which takes many cycles and the number of cycles is dependent on the data, which is vulnerable to side-channel timing attacks. Montgomery modular reduction performs the reduction using only multiplications in fixed time.

Naive remainder definition:

$$reduce(x) = x - floor(x/n) * n$$

Montgomery modular reduction (REDC):

$$reduce(x) * invR = x - (x * invN \ mod \ R) * N/R$$

where:

- $R = 2^{32}$ the register modulus
- $invN = N^{-1} mod \ R$
- $invR = R^{-1} mod \ N$

Pseudocode:

```
uint64_t t1 = x;
uint64_t t2 = (uint64_t)((uint32_t)x*invN)*N;
x = (uint32_t)((t1-t2)>>32);
x += N & -(t1<t2);
```

In other words, the modular reduction divides by R as a side effect (in the form of shift right 32):

$$REDC(x) = x * invR \ mod \ N$$

To perform a multiplication x=a*b, we must first convert a & b to Montgomery space:

$$mont(a) = a * R \ mod \ N$$
$$mont(b) = b * R \ mod \ N$$

But this again involves more modular reductions on secret data (on which no DIV instruction is allowed), which can be performed using more Montgomery multiplications:

$$REDC\big(a * (R^2 mod \ N)\big) = a * R \ mod \ N$$
$$REDC\big(b * (R^2 mod \ N)\big) = b * R \ mod \ N$$

where $R^2 mod \ N$ is precomputed.

After converting all inputs to Montgomery space, any number of multiplications can be performed correctly:

$$REDC(a * b) = a * b * R \ mod \ N$$

And after finishing the exponentiation by repeated squaring algorithm, the result is converted back from Montgomery to normal space:

$$REDC(a * R) = a$$

Resources:

https://en.algorithmica.org/hpc/number-theory/montgomery/