CS575: Final Project Report

Algorithms for Accelerated RSA Encryption Implementation

Team Member(s): Ritik Ramaiya

I. Problem

RSA is one of the first practical public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private). In the RSA, this asymmetry is based on the practical difficulty of the factorization of the product of two large Co-prime numbers, the "factoring problem". The project aims to describe the most popular algorithms to accelerate RSA implementations in software: Sliding window exponentiation, long number modular multiplication using the Montgomery algorithm, and the Karatsuba method for fast multiplication.

II. Algorithms

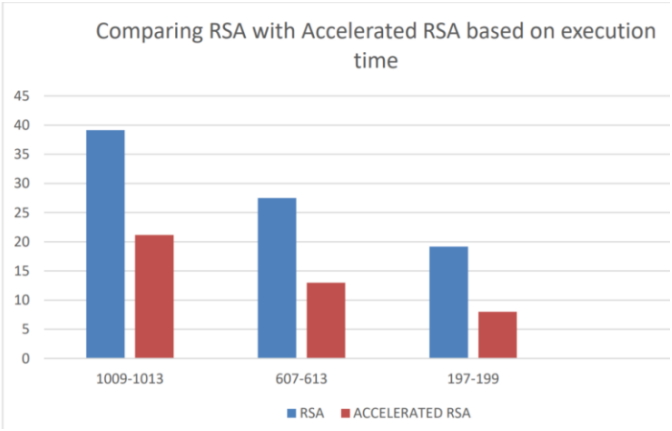
KARATSUBA METHOD:
Method: Let x and y be represented as n-digit strings in some base B. For any positive integer m
less than n, one can write the two given numbers as
x = x 1 B^m + x0,y = y 1 B^m + y0
xy = z2B^2m + z1B^m + z0
z2 = x1 y1, z1 = x1 y0 + x0 y1, z0 = x0 y0.
These formulae require four multiplications. Karatsuba observed that xy can be computed in only three multiplications, at the cost of a few extra additions. With z0 and z2 as before one can
A more efficient implementation of Karatsuba multiplication can be set as
x y = (b2 + b) x1y1 – b (x1 - x0) (y1 - y0) + (b + 1) x0 y0
where b = B^m

MONTGOMERY ALGORITHM:
Let T be an integer and choose an R>N such that gcd (R, N) =1 The Montgomery reduction of T is defined as: TR-1 (mod N) let m= T x (-N-1) (mod R) and t=(T+mN)/R thus, we have tR (mod N) = T (mod N) thus, t (mod N) = TR-1 (mod N) so, TR-1 (mod N) = (T + (T x (-N-1) (mod R) *N))/R.
Thus, to compute a x b (mod N)
we compute:
1. (-N-1) (mod R)
2. a'=aR (mod N)
3. b'=bR (mod N)
4. c'= (a'b') R-1 (mod N)
5. c= c' R-1 (mod N)

III. Software Design and Implementation

The algorithms have been implemented using C++ language and compiled using codeblocks:IDE which is a simple compiler for C++ codes.

Performance Evaluation



It was observed that by the utilisation of various methods the amount of computation for RSA can be severely reduced since it relies heavily on specific operations such as modular exponentiation and multiplication, thus any simplification of these operations leads to a considerable gain in efficiency for this algorithm that is sure to get faster in the future where it will continue to be useful for encryption and we observed that as we take large number as prime numbers the execution time will also be increased.

IV. Project Outcomes

- https://youtu.be/h7QHLrj2Xxk
- https://docs.google.com/presentation/d/1WSUp7bidUjtvkLhrr-mMJa6xmS6y7onX/edit?usp=sharing&ouid=101235259757921245225&rtpof=true&sd=true

References

[1]"Fast Pre-Processing For The Sliding Window Method Using Genetic Algorithms" – Nadia Nedjah and Luiza de Macedo Mourelle - Department of Systems Engineering

[2]“Window Method Using Genetic Algorithms” – sarah and Luiza de Macedo Mourelle - Department of IT

[3] Fast Montgomery Modular Multiplication and RSA Cryptographic Processor Architectures by Ciaran McIvor, Máire McLoone, John V McCanny

[4] Karatsuba Algorithm Based Accelerator for Pairing Computation by Yi Wu; GuoQiang Bai; XingJun Wu

[5] RNS Montgomery reduction algorithms using quadratic residuosity By Shinichi Kawamura, Hideo Shimizu