CHAPTER 4

MODULAR MULTIPLICATION METHODS

4.1 INTRODUCTION

Typically, public key cryptographic systems consist of (i) raising elements of some group such as GF (2^n), Zero / Non Zero (Z/NZ) or elliptic curves, to large powers and (ii) reducing the result modulo some given element. The above two operations are called modular exponentiation and can be performed using repeated modular multiplications.

Consequently the efficiency of Cryptographic systems depends heavily on how fast and efficiently modular exponentiations are performed.

The modular exponentiation is a key technique for scrambling and is used in several public-key cryptosystems. Therefore, improving the efficiency of modular multiplication will significantly enhance the efficiency of many cryptographic algorithms. As the bit length of numbers typically used in cryptographic applications is large, operations between these multi-precision numbers have to be mapped to operations between single digits. Of these digit operations the most expensive one (in terms of computation time) is the digit multiplication and usually, the number of digit multiplications is considered as the complexity metric.
4.2 MODULAR MULTIPLICATION ALGORITHMS

Modular multiplication is a complicated arithmetic operation due to the inherent multiplication and division operations involved. There are two main approaches to computing modular multiplication: (1) Perform the modulo operation after multiplication or (2) during multiplication. The modulo operation is accomplished by integer division in which, only the remainder is retained for further computation. The first approach requires a \( n \times n \) bit multiplier with a \( 2n \)-bit register followed by a \( 2n / n \) bit divider. In the second approach, the modulo operation occurs in each iteration step of integer multiplication. The advantage of the first method is that one can use any on the shelf method for multiplication and reduction. The main disadvantage of the second method is that it requires a great deal of space to store data, for further use by the reduction step. Hence, the first approach requires more hardware, while the second involves more addition/subtraction computations. The multiply then reduce methods consist of first computing the product and then reducing it with respect to the given modulus. These techniques are very fast and are suitable for hardware implementations. There are various algorithms to perform modular multiplications. The most prominent ones are

- Karatsuba – Ofman Algorithm.
- Booth Algorithm.
- Barrett Reduction Method.
- Montgomery Algorithm.

4.3 MONTGOMERY MULTIPLICATION

This algorithm computes the product of two integers modulo a third one without performing division by \( M \). It yields the reduced product using a series of additions.
4.4 BARRETT’S REDUCTION

Barrett P. introduced the idea of estimating the quotient $x \div m$ with operations that either are less expensive in time than a multiprecision division by $m$ (viz., 2 divisions by a power of $b$ and a partial multiprecision multiplication), or can be done as a precalculation for a given $m$ (viz., $\mu = b^{2k}/m$, i.e., $\mu$ is a scaled estimate of the modulus’ reciprocal). The estimate $q$ of $x \div m$ is obtained by replacing the floating point divisions in $q = (x/b^{2k}) (b^{2k}/m)/b'$ by integer divisions.

4.5 MODIFIED BOOTH RECODING

This converts the multiplier into higher radix numbers thereby reducing the number of partial products to be added, by half. The scheme is modified from the original Booth’s recoding to avoid a variable-size partial product array.

4.6 KARATSUBA OFMAN ALGORITHM

Karatsuba-Ofman algorithm is considered as one of the fastest ways to multiply long integers. It is based on a divide and conquer strategy. A multiplication of $2n$ digit integer is reduced to two $n$ digit multiplications, one $(n+1)$ digit multiplication, two $n$ digit subtractions, two left shift operations, two $n$ digit additions and two $2n$ digit additions. The algorithm can be explained as follows:

Let $X$ and $Y$ are the binary representation of two long integers given by equation (4.1).
\[ X = \sum_{i=0}^{n-1} X_i 2^i \quad \text{and} \quad Y = \sum_{i=0}^{n-1} Y_i 2^i \]  

(4.1)

where \(X_i, Y_i\) are \(i^{th}\) bits of \(X\) and \(Y\). \(2^i\) shows the weight of the bit.

Then to compute the product \(XY\) using the divide and conquer strategy, the operands \(X\) and \(Y\) can be decomposed into equal size parts \(X_H\) and \(X_L\), \(Y_H\) and \(Y_L\) as in eqn. 4.2 and , where subscripts H and L represent high and low order bits of \(X\) and \(Y\) respectively.

Let \(k = 2n = \) the number of bits in an operand. If \(k\) is odd, it can be right padded with a zero.

\[ X = 2^n \left( \sum_{i=0}^{n-1} X_i 2^i \right) + \sum X_i 2^i = X_H 2^n + X_L \]  

(4.2)

\[ Y = 2^n \left( \sum_{i=0}^{n-1} Y_i 2^i \right) + \sum Y_i 2^i = Y_H 2^n + Y_L \]  

(4.3)

The product \(XY\) can be computed as follows:

\[
P = X \cdot Y = (X_H 2^n + X_L)(Y_H 2^n + Y_L) \]

\[
= 2^n (X_H Y_H) + 2^n ((X_H Y_L) + (X_L Y_H)) + (X_L Y_L)
\]

(4.4)

From equation (4.2), it is obvious that the standard multiplication algorithm needs four \(n\) bits multiplications to compute the product \(P\). Nedjah N. et al. (2006) has proved that, the total number of one bit operations required using a standard multiplication algorithm is \(T(k) = (K \log_2 4) = K^2\).

The computation \(P\) can be further improved by observing that

\[
X_H Y_L + X_L Y_H = (X_H + X_L)(Y_H + Y_L) - X_H Y_H - X_L Y_L
\]

(4.5)
It can be shown that the 2n bits multiplication can be reduced to three n bits multiplications, namely $X_H Y_H$, $X_L Y_L$ and $(X_H+X_L)(Y_H+Y_L)$. This algorithm requires the total number of one bit operations as follows:

$$T(k) = (K \log_2 \frac{3}{2}) = k^{1.58}$$ (4.6)

Hence this algorithm is asymptotically faster than the standard multiplication algorithm.

4.7 SUMMARY

The different modular multiplication methods existing are listed in this chapter and the Karatsuba Ofman method is explained in detail. This has many applications in cryptographic key generation applications and digital signal processing applications. Hence this is considered in this work for implementation using Vedic Mathematics.