Spectral Modular Exponentiation Architecture for Public-Key Cryptography

Çetin Kaya Koç and Gökay Saldamlı
1250 NW 17th Street, Corvallis, Oregon 97331, USA
Email: koc@cryptocode.net
Phone: +1 541 908 3711

U.S. Patent Application - April 22, 2005

Abstract

We describe a new hardware architecture to perform the modular exponentiation operation, i.e., the computation of $c = m^e \bmod n$ where c, m, e, n are large integers. The modular exponentiation operation is the most common operation in public-key cryptography. The new method, named the Spectral Modular Exponentiation method, uses the Discrete Fourier Transform over a finite ring, and relies on new techniques to perform the modular multiplication and reduction operations. The method yields an efficient and highly parallel architecture for hardware implementations of public-key cryptosystems which use the modular exponentiation operation as the basic step, such as the RSA and Diffie-Hellman algorithms. The method with small modifications can also be used to compute the elliptic curve point multiplication operation in elliptic curves defined over the Galois field $GF(n)$ where n is a prime number. The details of elliptic curve computations and the hardware architecture are also described in this document.

1 Introduction

The Spectral Modular Exponentiation (SME) method is a new method for modular exponentiation of large integers. It takes the integers m, e, n as input, and computes c as the output such that

$$c = m^e \pmod{n} \ .$$

The exponent e can be of any length, and we assume that it is a j-bit integer such that $j \geq 128$. On the other hand, the integers c, m, n are k-bit integers with $k \geq 512$. It is assumed that $k = su$ for some integers s and u, where u is the wordsize of the implementation, i.e., the number of bits in a single word. A k-bit number is viewed as a vector of s words such that each word is u bits. Let a_i be the ith word of the integer a, then the vector representation means that we express the integer in the radix $b = 2^u$ as

$$a = (a_{s-1} a_{s-2} \cdots a_1 a_0)_b$$

where $0 \leq a_i \leq b - 1$. This representation implies that

$$a = \sum_{i=0}^{s-1} a_i b^i = \sum_{i=0}^{s-1} a_i 2^{ui} = a_0 + a_1 2^u + a_2 2^{2u} + \cdots + a_{s-1} 2^{(s-1)u} \ .$$

We will also treat integers as polynomials with an indeterminate t. The value of the integer is obtained from its polynomial by evaluating it at $t = b$. The above integer can be written in the polynomial notation as

$$a(t) = \sum_{i=0}^{s-1} a_i t^i = a_0 + a_1 t + a_2 t^2 + \cdots + a_{s-1} t^{(s-1)} \ .$$

We perform our arithmetic in the finite integer ring Z_q. If q is a prime, then the underlying structure will be a finite field. The arithmetic of this ring or field is simply modulo q arithmetic. Another important assumption we make in the SME method is that a d-point DFT exists over the ring Z_q. This assumption requires that

- The multiplicative inverse of d exists in Z_q, which requires $\gcd(d, q) = 1$.
- A principal dth root of unity exists in Z_q, which requires that d divides $p - 1$ for every prime p divisor of q. We will denote the principal dth root of unity with w.

We will use the notation

$$A(t) = \mathrm{DFT}_d^w[a(t)]$$

to denote the d-point DFT using the dth root of w, transforming the time-domain polynomial $a(t)$ to the spectral-domain polynomial $A(t)$. The d-point DFT is performed in the finite ring Z_q. Since q, d, and w are fixed, we will also use the simpler notation $A(t) = \mathrm{DFT}[a(t)]$ to denote the DFT operation. If a has s words in it, then, its polynomial will also have s words. In general, the value of d is fixed by the ring, and the value of s is equal to $s = \lceil d/2 \rceil$. The empty places in the polynomial $a(t)$, which are the coefficients of the higher orders of t, are filled with zeros in $a(t)$ before applying the d-point DFT. Inevitably, we also need the inverse DFT function, which we denote using the notation

$$a(t) = \mathrm{IDFT}_d^w[A(t)]$$

or simply as $a(t) = \mathrm{IDFT}[A(t)]$. We also make use of the following vector in our operations

$$\Gamma = [1, w^{-1}, w^{-2}, \ldots, w^{-(d-1)}] \ .$$

The polynomial representation of Γ is given as

$$\Gamma(t) = 1 + w^{-1} t + w^{-2} t^2 + \cdots + w^{-(d-1)} t^{d-1} \ .$$

The modulus n is a special number, for example, it is the product of two primes in RSA, and a prime number in Diffie-Hellman. Our assumption is that n is always odd. Let n_0 denote the least significant word of n, i.e., $n_0 = n \pmod b$, which is also an odd integer. Let ν_0 be the multiplicative inverse of n_0 modulo b, i.e.,

$$\nu_0 = n_0^{-1} \pmod b \ .$$

We need to use an integer multiple of n in our method, which we denote by θ, and is derived from n such that

$$\theta = \nu_0 n \ .$$

Note that since $\nu_0 n_0 = 1 \pmod b$, the least significant word of θ is equal to 1, i.e., $\theta_0 = 1$. The inverse $\nu_0 = n_0^{-1} \pmod b$ can be computed using the extended Euclidean algorithm, and we will give an explicit algorithm to compute it in this paper.

Let $\lambda = b^d \pmod{n}$. We name this integer as the Montgomery coefficient, since it is used inside the Montgomery multiplication algorithm. The number b^d is a $2s$-word number, however, λ is an s-word number since $\lambda = b^d \pmod{n}$. We will also use the polynomial representation of λ, which is denoted by $\lambda(t)$. Furthermore, we also define δ as the square of λ modulo n. The number δ is also an s-word number since $\delta = \lambda^2 \pmod{n}$. We will also the polynomial representation of δ, which is denoted by $\delta(t)$.

In addition to the usual scalar multiplication, we will also utilize the componentwise multiplication of vectors in the ring Z_q, and denote this operation with the symbol $\odot$ as

$$c(t) = a(t) \odot b(t) \ .$$

Given the vectors $a(t) = (a_0, a_1, \ldots, a_{d-1})$ and $b(t) = (b_0, b_1, \ldots, b_{d-1})$, the resulting vector after the $\odot$ operation will be $c(t) = (c_0, c_1, \ldots, c_{d-1})$ such that $c_i = a_i b_i \pmod{q}$ for $i = 0, 1, \ldots, d-1$.

Table 1: The symbols used in the SME method.

Symbol	Meaning	Relationship
c, m, e, n	Output and input integers	$c = m^e \pmod{n}$
k	Number of bits in c, m, n	usually $k \geq 512$
u	Length of a single word	k is a multiple of u
s	Number of words in c, m, n	$k = su$
b	Radix of representation	$b = 2^u$
$a(t)$	Polynomial representation of a	$a(b) = a$
d	Length of the DFT	$s = \lceil d/2 \rceil$
Z_q	Ring of integers modulo q	$\gcd(d, q) = 1$
p	A prime which divides q	d divides $p-1$ for every p
w	Principal dth root of unity in Z_q	$w^d = 1 \pmod{q}$
$\mathrm{DFT}_d^w[a(t)]$	d-point DFT of $a(t)$ in Z_q	$A(t) = \mathrm{DFT}[a(t)]$
$\mathrm{IDFT}_d^w[A(t)]$	Inverse DFT function	$a(t) = \mathrm{IDFT}[A(t)]$
Γ	$\Gamma = [1, w^{-1}, w^{-2}, \ldots, w^{-(d-1)}]$	w is dth root of unity
n	Modulus	n is odd
n_0	Least significant word of n	$n_0 = n \pmod{b}$
ν_0	Inverse of n_0 modulo b	$\nu_0 = n_0^{-1} \pmod{b}$
θ	ν_0 multiple of n	$\theta = \nu_0 n$ and $\theta_0 = 1$
λ	Montgomery coefficient	$\lambda = b^d \pmod{n}$
δ	Square of λ modulo n	$\delta = \lambda^2 \pmod{n}$
$\odot$	Componentwise multiplication in Z_q	$c(t) = a(t) \odot b(t)$

2 Spectral Modular Exponentiation Method

The Spectral Modular Exponentiation (SME) method relies on the Spectral Modular Multiplication (SMM) method, which is described in detailed in the following section. It also relies on the d-point Discrete Fourier Transform function in Z_q and the function $\mathrm{DFT}_d^w[.]$ needs to be available. However, the DFT function is used only 5 times in the SME method, and thus, the efficiency of the DFT implementation is not very crucial for the efficiency of the SME method.

Furthermore, any addition chain (exponentiation) method can be used in the SME method. We illustrate it using the binary method, however, more advanced algorithms such as the m-ary method, sliding windows method, etc., can also be utilized.

Input: The inputs are m, e, and n such that m and n are s words and the exponent e is j bits.

Preprocessing: It is often the case that the modulus n is available before the message m, for example, this is true for the digital signature algorithms. Therefore, we break the preprocessing into two stages as follows.

Preprocessing with n: Given n, obtain n_0.

1. Compute $\nu_0 = n_0^{-1} \pmod{2^u}$ using the extended Euclidean algorithm as follows:
   $\nu_0 = 1$
   for $i = 2$ **to** u
   if $n_0\nu_0 \geq 2^{i-1} \bmod 2^i$ **then** $\nu_0 = \nu_0 + 2^{i-1}$
   return ν_0
2. Compute $\theta = \nu_0 n$ and obtain the polynomial $\theta(t)$. Note that the least significant word of θ is 1, i.e., $\theta_0 = 1$.
3. Compute $\Theta(t) = \text{DFT}[\theta(t)]$ using the DFT function. An efficient implementation of the DFT function, i.e., the FFT (Fast Fourier Transform) algorithm, may be desired. However, simpler DFT implementations, for example, the matrix-vector product implementation can also be utilized. The DFT and the inverse DFT functions require d and w. Additionally, $d^{-1} \bmod q$ is precomputed and saved.
4. Compute the s-word integers $\lambda = b^d \pmod{n}$ and $\delta = \lambda^2 \pmod{n}$ using integer division. Also obtain the d-word polynomial representation $\delta(t)$ of δ. Again the higher order words are filled with zeros to make the polynomial d words. We then use the DFT function to compute $\Delta(t) = \text{DFT}[\delta(t)]$.
5. Assign $x = 1$ and obtain the polynomial representation of $x(t)$ which is equal to $x(t) = 1$. Compute the polynomial $X(t) = \text{DFT}[x(t)]$. This step is simplified using the fact that $x(t) = 1$, and thus, we do not need to use the DFT function to compute $X(t)$. It is obtained using assignment as $X(t) = 1 + 1 \cdot t + \ldots + 1 \cdot t^{d-1}$.

Preprocessing with m: Given m, obtain its polynomial representation $m(t)$.

1. Compute the polynomial $M(t) = \text{DFT}[m(t)]$.
2. Assign $C(t) = X(t)$. Recall that $x(t) = 1$ and $X(t) = \text{DFT}[x(t)] = [1, 1, \ldots, 1]$.
3. Use the Spectral Modular Multiplication (SMM) method to multiply $M(t)$ and $\Delta(t)$ in order to obtain $\overline{M}(t)$. We will denote this step by
   $$\overline{M}(t) = \text{SMM}[M(t), \Delta(t)] \ .$$
   The definition and detailed steps of the SMM method are given in the next section.
4. Use the SMM method to multiply $C(t)$ and $\Delta(t)$ in order to obtain $\overline{C}(t)$. We will denote this step by
   $$\overline{C}(t) = \text{SMM}[C(t), \Delta(t)] \ .$$

Exponentiation Loop: The exponentiation operation is performed as soon as the j-bit exponent e is available. Let the binary expansion of e be $(e_{j-1}e_{j-2}\cdots e_1e_0)_2$. The exponentiation operation needs $\overline{C}(t)$ and $\overline{M}(t)$ as input in addition to the exponent e.

for $i = j - 1$ **downto** 0
  $\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{C}(t)]$
  if $e_i = 1$ **then** $\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{M}(t)]$

Postprocessing: After the exponentiation loop is completed, we will have a final value of $\overline{C}(t)$. This vector will now be brought back to the time domain as follows.

1. Obtain $C(t)$ using the SMM method by multiplying $\overline{C}(t)$ and $X(t)$ as follows

$$C(t) = \text{SMM}[\overline{C}(t), X(t)] \ .$$

2. Obtain $c(t)$ using the Inverse DFT function as follows

$$c(t) = \text{IDFT}[C(t)] \ .$$

Output: The integer c representing the polynomial $c(t)$ is the output such that $c = m^e \pmod{n}$.

3 Spectral Modular Multiplication Method

The SMM method takes two arguments, such as $A(t)$ and $B(t)$, and computes $S(t)$ as the output. We will denote the operation using

$$R(t) = \text{SMM}[A(t), B(t)] \ .$$

The steps of the SMM method are given below.

1: $R(t) = A(t) \odot B(t) \pmod{q}$
2: $\alpha = 0$
3: **for** $i = 0$ **to** $d - 1$
4: $r_0 = d^{-1}(R_0 + R_1 + \cdots + R_{d-1}) \pmod{q}$
5: $\beta = -(r_0 + \alpha) \pmod{b}$
6: $\alpha = (r_0 + \alpha + \beta)/b$
7: $R(t) = R(t) + \beta \cdot \Theta(t) \pmod{q}$
8: $R(t) = R(t) - (r_0 + \beta) \pmod{q}$
9: $R(t) = R(t) \odot \Gamma(t) \pmod{q}$
10: **end for**
11: **return** $R(t)$

The steps of the SMM method are explained in detail below. In addition to the inputs, the SMM function has access to parameters available after the preprocessing steps of the SME method. These parameters are d^{-1}, $\Theta(t)$, and $\Gamma(t)$. The SME method performs only modulo q and modulo b operations.

1. This is the vector $\odot$ operation in the ring Z_q. Given the vectors $A(t)$ and $B(t)$, we compute $R(t)$ such that $R_i = A_i B_i \pmod{q}$ for $i = 0, 1, \ldots, d - 1$.

2. Initial value of α is assigned as zero.

3. The **for** loop is executed d times for $i = 0, 1, \ldots, d - 1$.

4. After step 1, we have the vector $R(t)$. The elements are summed modulo q to obtain

$$R_0 + R_1 + \cdots + R_{d-1} \pmod{q} ,$$

and multiplied by $d^{-1} \bmod q$ in order to obtain r_0. The inverse d^{-1} was already computed in the preprocessing stage of the SME method.

5. Since $b = 2^u$, the computation of $\beta = -(r_0 + \alpha) \bmod b$ is a 1-word operation, involving an addition and sign-change (2's complement) operations on 1-word numbers.

6. The value of α is updated for the next loop instance as $\alpha = (r_0 + \alpha + \beta)/b$.

7. The 1-word number β is multiplied with every element of $\Theta(t)$ and the result is added to the corresponding element of $R(t)$. Given $N(t) = (N_0, N_1, \ldots, N_{d-1})$, we compute the elements of the new $R(t)$ as

$$R_i = R_i + \beta \cdot \Theta_i \pmod{q}$$

for $i = 0, 1, \ldots, d-1$.

8. This is a vector operation. The 1-word number $r_0 + \beta$ is subtracted from every element of $R(t)$, and therefore, we have

$$R_i = R_i - (r_0 + \beta) \pmod{q}$$

for $i = 0, 1, \ldots, d-1$.

9. This is also a vector operation. The resulting $R(t)$ from the previous step is multiplied componentwise with the vector $\Gamma(t)$. Therefore, the elements of the new $R(t)$ are found as

$$R_i = R_i \cdot \Gamma_i \pmod{q}$$

for $i = 0, 1, \ldots, d-1$.

10. The end of **for** loop.

11. The multiplication result $R(t) = \mathrm{SMM}[A(t), B(t)]$ is returned.

4 An Example Exponentiation using the SME Method

In this section, we give an example exponentiation computation using the SME method with the input values as $m = 27182$, $e = 53$, and $n = 31417$. We will describe the steps of the SME method performing this modular exponentiation operation, giving the temporary results and the final result $c = m^e \pmod{n}$.

We select the length of a single word as $u = 4$. Therefore, the radix of the representation is $b = 2^u = 16$, i.e., the numbers are represented in hexadecimal. Since

$$n = (31417)_{10} = (0111\ 1010\ 1011\ 1001)_2 = (7AB9)_{16} ,$$

we have $k = 16$ and $s = 4$. The polynomial representation of n is given as

$$n(t) = 9 + 11t + 10t^2 + 7t^3 ,$$

in which the digits are expressed in decimal. The integer value $n = (31417)_{10} = (7AB9)_{16}$ is obtained by evaluating $n(t)$ at $t = 16$ in any selected basis.

We will perform our computations in the Fermat ring Z_q where $q = 2^{20} + 1$. Since $s = 4$, we need a DFT function in this ring with the length $d = 8$. It turns out that such DFT exists in this ring with the principal 8th root of unity given as $w = 32$. Furthermore, the vector $\Gamma(t)$ is given as

$$\begin{aligned}\Gamma(t) &= 1 + w^{-1}t + w^{-2}t^2 + w^{-3}t^3 + w^{-4}t^4 + w^{-5}t^5 + w^{-6}t^6 + w^{-7}t^7 \\ &= 1 + 1015809t + 1047553t^2 + 1048545t^3 + 1048576t^4 + \\ &\quad 32768t^5 + 1024t^6 + 32t^7 \ .\end{aligned}$$

The steps of the SME method computing this modular exponentiation are described below.

Preprocessing with n: Given $n = (7AB9)_{16}$, we have $n_0 = 9$.

1. The inversion algorithm computes $\nu_0 = 9^{-1} \pmod{16}$ as $\nu_0 = 9$.
2. The computation of $\theta = \nu_0 n$ gives $9 \cdot 31417 = 282753$, which is expressed in binary and in hexadecimal as

   $$\theta = (0100\ 0101\ 0000\ 1000\ 0001)_2 = (45081)_{16} \ .$$

   Recall that θ is $s + 1 = 5$ words and $\theta_0 = 1$. We also obtain the polynomial $\theta(t)$ as

   $$\theta(t) = 1 + 8t + 5t^3 + 4t^4 \ .$$

3. The computation of $\Theta(t) = \mathrm{DFT}[\theta(t)]$ is accomplished using the DFT function. In Section 4, we describe the DFT computation and particularly evaluate this transform. We obtain the result of the DFT as

   $$\begin{aligned}\Theta(t) &= 18 + 164093t + 3077t^2 + 262301t^3 + 1048569t^4 + \\ &\quad 884478t^5 + 1045510t^6 + 786270t^7 \ .\end{aligned}$$

   Recall that we work in the finite ring Z_q for $q = 2^{20} + 1 = 1048577$, and thus, the coefficients of the polynomial $\Theta(t)$ are in the range $[0, 2^{20} + 1)$.
   In this step, we also compute and save $d^{-1} \pmod{q}$ as

   $$d^{-1} = 8^{-1} \pmod{2^{20} + 1} = 917505 \ .$$

4. We compute the Montgomery coefficient and its square as

   $$\begin{aligned}\lambda &= 16^8 \pmod{31417} = 12060 \ , \\ \delta &= 12060^2 \pmod{31417} = 14307 \ .\end{aligned}$$

   The polynomial representation of δ is found using $\delta = (14307)_{10} = (37E3)_{16}$ as

   $$\delta(t) = 3 + 14t + 7t^2 + 3t^3 \ .$$

   Furthermore, we obtain the spectral representation of $\delta(t)$ using the DFT as

   $$\begin{aligned}\Delta(t) &= 27 + 105923t + 11260t^2 + 451683t^3 1048570t^4 + \\ &\quad 956996t^5 + 1037309t^6 + 582564t^7 \ .\end{aligned}$$

5. Given $x(t) = 1$, the spectral representation of $x(t)$ is found as

$$X(t) = 1 + t + t^2 + t^3 + t^4 + t^5 + t^6 + t^7 \ .$$

Preprocessing with m: Given $m = (27182)_{10} = (6A2E)_{16}$, we have $m(t) = 14 + 2t + 10t^2 + 6t^3$.

1. Given $m(t)$, we obtain its spectral representation $M(t)$ using the DFT as

$$\begin{aligned} M(t) &= 32 + 206926t + 1044485t^2 + 55502t^3 + 16t^4 \\ &\quad 862159t^5 + 4100t^6 + 972623t^7 \ . \end{aligned}$$

2. The initial value of $C(t)$ is given as

$$C(t) = X(t) = 1 + t + t^2 + t^3 + t^4 + t^5 + t^6 + t^7 \ .$$

3. The SMM method is used to compute $\overline{M}(t) = \text{SMM}[M(t), \Delta(t)]$ with inputs

$$\begin{aligned} M(t) &= 32 + 206926t + 1044485t^2 + 55502t^3 + 16t^4 + \\ &\quad 862159t^5 + 4100t^6 + 972623t^7 \ , \\ \Delta(t) &= 27 + 105923t + 11260t^2 + 451683t^3 + 1048570t^4 + \\ &\quad 956996t^5 + 1037309t^6 + 582564t^7 \ . \end{aligned}$$

We then use the SMM method to find the resulting polynomial $\overline{M}(t)$ given the inputs $M(t)$ and $\Delta(t)$. First we execute Step 1 in the SMM method, and obtain the initial value of $R(t)$ using the rule $R_i = M_i \cdot \Delta_i \pmod q$ for $i = 0, 1, \ldots, 7$ as

$$\begin{aligned} R(t) &= 864 + 866244t + 61468t^2 + 979527t^3 + 1048465t^4 + \\ &\quad 464721t^5 + 987165t^6 + 834767t^7 \ . \end{aligned}$$

In Step 2 of the SMM method, We assign the initial value of $\alpha = 0$, and start the **for** loop for $i = 0, 1, \ldots, 7$. We illustrate the computation of the instance of the loop for $i = 0$ in **Table 2**. The **for** loop needs to execute for the remaining values of i as $i = 1, 2, \ldots, 7$ in order to compute the resulting product $\overline{M}(t)$ which is found as

$$\begin{aligned} \overline{M}(t) &= 354 + 463771t + 11385t^2 + 686651t^3 + \\ &\quad 156t^4 + 722398t^5 + 1037434t^6 + 225086t^7 \ . \end{aligned}$$

Table 2: The SME method **for** loop instance $i = 0$.

Step	Operation and Result
4:	$r_0 = d^{-1} \cdot (R_0 + R_1 + R_2 + R_3 + R_4 + R_5 + R_6 + R_7) \pmod q$ $r_0 = 917505 \cdot (864 + 866244 + 61468 + 979527 + 104846 +$ $464721 + 987165 + 834767) \pmod{1048567} = 42$
5:	$\beta = -(r_0 + \alpha) \pmod b = -(42 + 0) \pmod{16} = 6$
6:	$\alpha = (r_0 + 0 + \beta)/b = (42 + 6)/16 = 3$
7:	$R_i = R_i + \beta \cdot \Theta_i \pmod q$ $R(t) = 972 + 802225t + 79930t^2 + 456179t^3 + 1048417t^4 +$ $528704t^5 + 968763t^6 + 309502t^7$
8:	$R_i = R_i - (r_0 + \beta) = R_i - 48 \pmod q$ $R(t) = 924 + 802177t + 79882t^2 + 456131t^3 + 1048369t^4 +$ $528656t^5 + 968715t^6 + 309454t^7$
9:	$R_i = R_i \cdot \Gamma_i \pmod q$ $R(t) = 924 + 802177t + 79882t^2 + 456131t^3 + 1048369t^4 +$ $528656t^5 + 968715t^6 + 599495t^7$

4. In this step, the SMM method is used to compute $\overline{C}(t) = \text{SMM}[C(t), \Delta(t)]$ with inputs

$$\begin{aligned} C(t) &= 1 + t + t^2 + t^3 + t^4 + t^5 + t^6 + t^7 \ , \\ \Delta(t) &= 27 + 105923t + 11260t^2 + 451683t^3 + 1048570t^4 + \\ &\quad 956996t^5 + 1037309t^6 + 582564t^7 \ . \end{aligned}$$

We will not give the details of this multiplication since it is similar to the previous one. The result is obtained as

$$\begin{aligned} \overline{C}(t) &= 342 + 472129t + 17495t^2 + 875041t^3 + \\ &\quad 132t^4 + 730372t^5 + 1031256t^6 + 20260t^7 \ . \end{aligned}$$

Exponentiation Loop: The loop starts with the values of $\overline{M}(t)$ and $\overline{C}(t)$ computed above as

$$\begin{aligned} \overline{M}(t) &= 354 + 463771t + 11385t^2 + 686651t^3 + \\ &\quad 156t^4 + 722398t^5 + 1037434t^6 + 225086t^7 \ . \\ \overline{C}(t) &= 342 + 472129t + 17495t^2 + 875041t^3 + \\ &\quad 132t^4 + 730372t^5 + 1031256t^6 + 20260t^7 \ . \end{aligned}$$

Given the exponent value $e = (53)_{10} = (110101)_2$, the exponentiation algorithm performs squarings and multiplications using the SMM method. Since $j = 6$, the value of i starts from $i = 5$ and moves down to zero, and computes the new value of $\overline{C}(t)$ using the binary method of exponentiation as described. The steps of the exponentiation and intermediate values of $\overline{C}(t)$ are tabulated in **Table 3**. The final value is computed as

$$\begin{aligned} \overline{C}(t) &= 123 + 34099t + 40979t^2 + 229426t^3 + \\ &\quad 43t^4 + 31539t^5 + 1007636t^6 + 753717t^7 \ . \end{aligned}$$

Table 3: The steps of the exponentiation loop.

i	e_i	Operation	$\overline{C}(t)$
		Start	$342 + 472129t + 17495t^2 + 875041t^3 + 132t^4 + 730372t^5 + 1031256t^6 + 20260t^7$
5		$\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{C}(t)]$	$270 + 44476t + 108628t^2 + 286841t^3 + 58t^4 + 37692t^5 + 940117t^6 + 680064t^7$
	1	$\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{M}(t)]$	$288 + 741281t + 71696t^2 + 769759t^3 + 68t^4 + 473378t^5 + 976913t^6 + 113124t^7$
4		$\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{C}(t)]$	$348 + 869774t + 81984t^2 + 183179t^3 + 92t^4 + 338831t^5 + 966721t^6 + 705938t^7$
	1	$\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{M}(t)]$	$297 + 42796t + 20613t^2 + 615596t^3 + 129t^4 + 39470t^5 + 1028230t^6 + 351407t^7$
3		$\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{C}(t)]$	$123 + 34099t + 40979t^2 + 229426t^3 + 43t^4 + 31539t^5 + 1007636t^6 + 753717t^7$
	0		$123 + 34099t + 40979t^2 + 229426t^3 + 43t^4 + 31539t^5 + 1007636t^6 + 753717t^7$
2		$\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{C}(t)]$	$336 + 857269t + 74835t^2 + 1014675t^3 + 94t^4 + 326774t^5 + 973908t^6 + 947609t^7$
	1	$\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{M}(t)]$	$183 + 162592t + 51267t^2 + 691423t^3 + 67t^4 + 945569t^5 + 997444t^6 + 297954t^7$
1		$\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{C}(t)]$	$348 + 869774t + 81984t^2 + 183179t^3 + 92t^4 + 338831t^5 + 966721t^6 + 705938t^7$
	0		$348 + 869774t + 81984t^2 + 183179t^3 + 92t^4 + 338831t^5 + 966721t^6 + 705938t^7$
0		$\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{C}(t)]$	$297 + 42796t + 20613t^2 + 615596t^3 + 129t^4 + 39470t^5 + 1028230t^6 + 351407t^7$
	1	$\overline{C}(t) = \text{SMM}[\overline{C}(t), \overline{M}(t)]$	$123 + 34099t + 40979t^2 + 229426t^3 + 43t^4 + 31539t^5 + 1007636t^6 + 753717t^7$

Postprocessing: After the exponentiation loop is completed, we have the final value $\overline{C}(t)$. In this step, we have two consecutive SMM executions.

- We obtain $C(t)$ using $C(t) = \text{SMM}[\overline{C}(t), X(t)]$ using the inputs

$$\begin{aligned} \overline{C}(t) &= 123 + 34099t + 40979t^2 + 229426t^3 + \\ &\quad 43t^4 + 31539t^5 + 1007636t^6 + 753717t^7 \ . \\ X(t) &= 1 + t + t^2 + t^3 + t^4 + t^5 + t^6 + t^7 \ . \end{aligned}$$

  This computation finds $C(t)$ as

$$C(t) = 312 + 76043t + 36932t^2 + 58506t^3 +$$

$$112t^4 + 71693t^5 + 1011781t^6 + 842895t^7 \ ,$$

- We obtain $c(t)$ using the inverse DFT function $c(t) = \text{IDFT}[C(t)]$, which gives

$$c(t) = 9 + 4t + 3t^2 + 6t^3 \ .$$

Thus, we obtain the final value as $c = (6349)_{16} = (25417)_{10}$, which is equal to

$$25417 = 227182^{53} \pmod{31417}$$

as required.

5 Computation of Discrete Fourier Transform in Z_q

The DFT of a sequence $a = [a_0, a_1, \ldots, a_{d-1}]$ is defined as the sequence $A = [A_0, A_1, \ldots, A_{d-1}]$ such that

$$A_j = \sum_{i=0}^{d-1} a_i w_d^{ij} \pmod{q} \ ,$$

where w is the dth root of unity. The DFT function is normally used over the field of complex numbers $\mathbf{C}$, however, in our application, we need a finite ring or field since we cannot perform infinite precision arithmetic.

The above sum can also be written as a matrix-vector product as

$$\begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ \vdots \\ A_{d-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{d-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(d-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & w^{d-1} & w^{2(d-1)} & \cdots & w^{(d-1)(d-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{d-1} \end{bmatrix} \pmod{q} \ .$$

We denote this matrix-vector product by $A = Ta$, where T is the $d \times d$ transformation matrix. The inverse DFT is defined as $a = T^{-1}A$. It turns out that the inverse of T is obtained by replacing w with w^{-1} in the matrix, and by placing the multiplicative factor d^{-1} in front of the matrix. The inverse matrix is given as

$$T^{-1} = d^{-1} \cdot \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w^{-1} & w^{-2} & \cdots & w^{-(d-1)} \\ 1 & w^{-2} & w^{-4} & \cdots & w^{-2(d-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & w^{-(d-1)} & w^{-2(d-1)} & \cdots & w^{-(d-1)(d-1)} \end{bmatrix} \pmod{q} \ .$$

The matrix-vector product definition of the DFT implies an algorithm to compute the DFT function, however, it requires d multiplications and $d-1$ additions to compute an entry of the output sequence A. Thus, the total number of multiplications is d^2, and the total number of additions is $d(d-1)$. This complexity is acceptable for the SME method since we need to use the DFT or IDFT functions only 4 times. If desired, the Fast Fourier Transform (FFT) algorithm can be used which reduces the complexity from $O(d^2)$ to $O(d \log d)$.

We now derive the transformation and inverse transformation matrices for the example DFT in Section 4, and perform a DFT computation for illustrating the properties of the arithmetic of the DFT in Z_q. In our example, we have $q = 2^{20} + 1$, $d = 8$, and $w = 32$. The elements of the transformation matrix T are of the form $w^{i \cdot j} \pmod q$ for $i, j = 0, 1, \ldots, 7$. We compute these values and place it in the transformation matrix as follows:

$$T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 32 & 1024 & 32768 & 1048576 & 1048545 & 1047553 & 1015809 \\ 1 & 1024 & 1048576 & 1047553 & 1 & 1024 & 1048576 & 1047553 \\ 1 & 32768 & 1047553 & 32 & 1048576 & 1015809 & 1024 & 1048545 \\ 1 & 1048576 & 1 & 1048576 & 1 & 1048576 & 1 & 1048576 \\ 1 & 1048545 & 1024 & 1015809 & 1048576 & 32 & 1047553 & 32768 \\ 1 & 1047553 & 1048576 & 1024 & 1 & 1047553 & 1048576 & 1024 \\ 1 & 1015809 & 1047553 & 1048545 & 1048576 & 32768 & 1024 & 32 \end{bmatrix} .$$

The inverse transformation matrix T^{-1} can be obtained similarly using the multiplicative inverses

$$\begin{aligned} d^{-1} &= 8^{-1} \pmod{2^{20}+1} = 917505 . \\ w^{-1} &= 32^{-1} \pmod{2^{20}+1} = 1015809 . \end{aligned}$$

We obtained it as

$$T^{-1} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1015809 & 1047553 & 1048545 & 1048576 & 32768 & 1024 & 32 \\ 1 & 1047553 & 1048576 & 1024 & 1 & 1047553 & 1048576 & 1024 \\ 1 & 1048545 & 1024 & 1015809 & 1048576 & 32 & 1047553 & 32768 \\ 1 & 1048576 & 1 & 1048576 & 1 & 1048576 & 1 & 1048576 \\ 1 & 32768 & 1047553 & 32 & 1048576 & 1015809 & 1024 & 1048545 \\ 1 & 1024 & 1048576 & 1047553 & 1 & 1024 & 1048576 & 1047553 \\ 1 & 32 & 1024 & 32768 & 1048576 & 1048545 & 1047553 & 1015809 \end{bmatrix} .$$

We will now perform the DFT computation $\Theta(t) = \mathrm{DFT}[\theta(t)]$ with the input polynomial

$$\theta(t) = 1 + 8t + 5t^3 + 4t^4 ,$$

as mentioned in Section 4. This polynomial is expressed as a vector

$$\theta = [1, 8, 0, 5, 4, 0, 0, 0]$$

and enters the DFT function, producing the output vector Θ. Using the matrix-vector product algorithm, we obtain the vector Θ as

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 32 & 1024 & 32768 & 1048576 & 1048545 & 1047553 & 1015809 \\ 1 & 1024 & 1048576 & 1047553 & 1 & 1024 & 1048576 & 1047553 \\ 1 & 32768 & 1047553 & 32 & 1048576 & 1015809 & 1024 & 1048545 \\ 1 & 1048576 & 1 & 1048576 & 1 & 1048576 & 1 & 1048576 \\ 1 & 1048545 & 1024 & 1015809 & 1048576 & 32 & 1047553 & 32768 \\ 1 & 1047553 & 1048576 & 1024 & 1 & 1047553 & 1048576 & 1024 \\ 1 & 1015809 & 1047553 & 1048545 & 1048576 & 32768 & 1024 & 32 \end{bmatrix} \begin{bmatrix} 1 \\ 8 \\ 0 \\ 5 \\ 4 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 18 \\ 164093 \\ 3077 \\ 262301 \\ 1048569 \\ 884478 \\ 1045510 \\ 786270 \end{bmatrix} .$$

Recall that all multiplications and additions are performed modulo q where $q = 2^{20}+1$. We express Θ as a polynomial

$$\begin{aligned}\Theta(t) &= 18 + 164093t + 3077t^2 + 262301t^3 + 1048569t^4 + \\ &\quad 884478t^5 + 1045510t^6 + 786270t^7\end{aligned}$$

which was the result obtained in Step 4 of 'Preprocessing with n' in Section 4.

6 Parameter Selection

In this section, we describe the methodology for selecting the parameters for the DFT and SME functions in order to apply the method in public-key cryptography. We will tabulate some example parameters for modular exponentiations using the SME method, starting from 530 bits up to 4,110 bits. We also tabulate typical rings and their DFT parameters for use in the SME function.

A ring for which q is of the form $2^v \pm 1$ is the most suitable for the SME computation since the modular arithmetic operations for such q are simplified. The rings of the form $2^v - 1$ are called the *Mersenne rings*, while the rings of the form $2^v + 1$ are called the *Fermat rings*. In **Table 8** and **Table 9** in the Appendix Section of this document, we tabulate the Fermat and Mersenne rings suitable for the SME function. Furthermore, we also tabulate a root of unity and the DFT length for each ring. Whenever possible, we select the root of unity as $w = 2$ or $w = -2$ since multiplication with such numbers are accomplished by shifting.

The Mersenne and Fermat rings are not the only suitable rings for the SME method. Let q'' (not necessarily a prime) be a small divisor of q'. The rings of the form $Z_{q'/q''}$ are also quite useful. Since q'' divides q', the arithmetic modulo (q'/q'') can be carried in the ring $Z_{q'}$. By selecting $Z_{q'}$ as a Mersenne or Fermat ring, we also simplify the arithmetic. Such rings are called *pseudo Mersenne* or *pseudo Fermat rings*. We also propose the use of such rings in the SME method. In **Table 10A, 10B, 10C** and **Tables 11A, 11B**, we tabulate the pseudo Mersenne and Fermat rings together with their root of unity and the length of the DFT values.

Once the underlying ring and the DFT length and the root of unity are selected, the maximum modulus size in the SME method can be computed by finding the radix b. The relation between these parameters is given by the following inequality

$$b^4 < \frac{3q}{s^3} \ . \tag{1}$$

In **Table 4**, we give some example rings and their parameters. To illustrate the methodology, we select a ring from this table, e.g., $q = q'/q'' = (2^{57} - 1)/7$. This ring comes with the root of unity $w = -2$ and the length $s = 57$. Using the q and s values and the above inequality, we compute

$$b < \sqrt[4]{\frac{3 \cdot (2^{57} - 1)}{7 \cdot (57)^3}} \approx 759.93 \ .$$

Therefore, we find $u = \lfloor \log_2(b) \rfloor = 9$. Since $s = 57$, we find the maximum bit length of the exponentiation as $k = s \cdot u = 57 \cdot 9 = 513$ as given in **Table 4**.

Table 4: Parameter selection.

Bits k	Ring q	DFT d	Root w	Wordsize u	Words s
513	$(2^{57}-1)/7$	114	-2	10	53
518	$2^{73}-1$	73	2	14	37
518	$(2^{73}+1)/3$	73	4	14	37
768	$2^{64}+1$	128	2	12	64
1,185	$2^{79}-1$	158	-2	15	79
1,792	$2^{128}+1$	128	2	28	64
2,060	$(2^{103}+1)/3$	206	2	20	103
2,163	$2^{103}-1$	206	-2	21	103
3,456	$(2^{128}+1)$	256	-2	27	128
4,170	$2^{139}-1$	274	-2	30	139

7 Improved Parameter Selection

The SMM method can be improved with a simple arrangement. It is possible to replace the multiplication $\beta \cdot \Theta(t)$ in Step 7 of the SMM method with additions at a cost of precomputations and storage space. This approach increases the wordsize and allows the selection of a smaller ring for a similar modulus size.

Let $\theta^i(t)$ be the polynomial representation of an integer multiple of n such that $\theta_0^i = 2^{i-1}$ for $i = 1, 2, \ldots, u$. We can now write $\beta \cdot \Theta(t)$ as

$$\beta \cdot \Theta(t) = \sum_{i=1}^{u} \beta_i \cdot \Theta^i(t) \ , \tag{2}$$

where β_i is a binary digit of β and $\Theta^i(t) = \mathrm{DFT}[\theta^i(t)]$ for $i = 1, 2, \ldots, u$. Note that $\beta < 2^u$ and $\beta_i = 0$ for $i \geq u$. The polynomial set $\{\Theta^1(t), \Theta^2(t), \ldots, \Theta^u(t)\}$ needs to be precomputed and stored. The precomputation can be done in the Preprocessing with n phase: Starting with multiplying $\theta^1(t)$ (which is $\theta(t)$ of the SMM) by powers of 2 after finding ν_0. We then apply the DFT function to these polynomials in order to get $\Theta^i(t) = \mathrm{DFT}[\theta^i(t)]$ for $i = 1, 2, \ldots, u$. With this adjustment a better bound for b could be given as

$$b^2 \log_2(b) < \frac{3q}{s^3} \ . \tag{3}$$

This bound gives us the improved parameters which are tabulated in **Table 5** below.

Table 5: Improved parameter selection.

Bits k	Ring q	DFT d	Root w	Wordsize u	Words s
540	$(2^{59}+1)/3$	59	2	19	30
564	$2^{47}-1$	94	-2	12	47
570	$2^{59}-1$	59	2	19	30
620	$2^{61}-1$	61	2	20	31
672	$2^{64}+1$	64	4	21	32
1,098	$2^{61}-1$	122	2	18	61
1,120	$2^{79}-1$	79	2	28	40
1,216	$2^{64}+1$	128	2	19	64
2,054	$2^{79}-1$	158	2	26	79
2,160	$2^{107}-1$	107	2	40	54
3,200	$2^{128}+1$	128	4	50	64
4,173	$2^{107}-1$	214	-2	39	107
6,272	$2^{128}+1$	256	2	49	128

Furthermore, in **Tables 4 & 5**, there are two important issues that should be considered for an efficient design. First issue is the number of words (s) or the DFT length (d) which are related to one another by $s = \lceil d/2 \rceil$. Observe that the maximum modulus size is given as $k = su$. Moreover, the loop in the SMM algorithm runs d times. Therefore, a decrease in d is desirable for some designs even it is at a cost of using larger rings (q) and requiring some more storage space. In **Table 6**, we demostrate parameters of this nature.

Table 6: Decreasing d.

Bits k	Ring q	DFT d	Root w	Wordsize u	Words s
540	$(2^{115}-1)/31$	23	32	45	12
560	$(2^{93}+1)/9$	31	64	35	16
608	$2^{96}+1$	32	64	38	16
1,024	$(2^{155}+1)/33$	31	1024	64	16
1,122	$(2^{129}+1)/9$	43	64	51	22
2,150	$(2^{129}+1)/9$	86	64	50	43

The second issue is the wordsize u which determines the maximum modulus size k as $k = su$ and the number of elements in the set $\{\Theta^1(t), \Theta^2(t), \ldots, \Theta^u(t)\}$ which needs to be stored. For instance, in the ring $q = 2^{47}-1$, the value of $u = 12$ implies that $\{\Theta^1(t), \Theta^2(t), \ldots, \Theta^{12}(t)\}$ need to be stored. Here $\Theta^i(t)$ is a sequence of length 86 whose elements are from the ring $q = 2^{47}-1$ for all $i = 1, 2, \ldots, 12$. This storage requirements of the above strategy can be excessive, however,

a hybrid stragedy is also possible. This can be summarized with the following equation

$$\beta \cdot \Theta(t) = \beta' \cdot \Theta^1(t) + \sum_{i=u'}^{u} \beta_i \cdot \Theta^i(t) \ ,$$

where $\beta' = \beta \bmod 2^{u'}$ and β_i stands for binary digits of $\beta \div 2^{u'}$ for some $0 \leq u' \leq u$.

8 Spectral Point Multiplication for Elliptic Curve Cryptography

An elliptic curve E over a prime field $GF(n)$ (i.e., n is a prime) is determined by parameters $a, b \in GF(n)$ which satisfy $4a^3 + 27b^2 \neq 0$. The curve consists of the set of solutions or points $\mathbb{P} = (x, y)$ for $x, y \in GF(n)$ to the equation

$$y^2 \equiv x^3 + ax + b \bmod n$$

together with an extra point $\mathbb{O}$ called the point at infinity. The set of points on E forms a group under the following addition rule: Let $(x_1, y_1) \in E(GF(n))$ and $(x_2, y_2) \in E(GF(n))$ be two points such that $x_1 \neq x_2$. Then, we have $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where

$$\begin{aligned} \lambda &= (y_2 - y_1)(x_2 - x_1)^{-1} \pmod n \ , \\ x_3 &= \lambda^2 - x_1 - x_2 \pmod n \ , \\ y_3 &= \lambda(x_1 - x_3) - y_1 \pmod n \ . \end{aligned}$$

All computations are performed within the finite field $GF(n)$. The security provided by ECC is guaranteed by the difficulty of the discrete logarithm problem in the elliptic curve group. The discrete logarithm problem is the problem of finding the least positive number, k, which satisfies the equation

$$\mathbb{P}_1 = e \times \mathbb{P}_0 = \underbrace{\mathbb{P}_0 + \mathbb{P}_0 + \cdots + \mathbb{P}_0}_{e \text{ times}} \ ,$$

where $\mathbb{P}_0$ and $\mathbb{P}_1$ are points on the elliptic curve. Naturally, the basic computation (called *point multiplication*) in ECC is finding the eth (additive) power of an element $\mathbb{P}_0$ in the group. This involves additions, multiplications, and inversions of integers which are in the coordinates of the points. That is, it relies completely upon calculations in the underlying field, $GF(n)$. But since the underlying field is a prime field the arithmetic is not different from the usual modular arithmetic. Therefore, the spectral methods can be employed for elliptic point multiplication likewise described for the modular exponentiation.

8.1 Modified Spectral Modular Multiplication Algorithm

It turns out that the least magnitude residue representation of integers in the ring Z_q is more suitable for applying the Spectral Modular Multiplication method in elliptic curve cryptography. We represent the integers in the ring Z_q with the set $\{-q/2, \ldots, -1, 0, 1, \ldots, q/2\}$. In this convention, the modular reduction method picks values from the least magnitude set, e.g., 12 mod 7 is equal to -2 instead of 5. We make some small changes in the SMM method in order to utilize the least magnitude residues properly. The modified spectral multiplication algorithm is denoted with the following operation

$$R(t) = \text{SMM2}[A(t), B(t)] \ .$$

The detailed steps of the SMM2 method are given below.

1: $R(t) = A(t) \odot B(t) \pmod q$
2: $\alpha = 0$
3: **for** $i = 0$ **to** $d - 1$
4: $r_0 = d^{-1}(R_0 + R_1 + \cdots + R_{d-1}) \pmod q$
5: $\beta = r_0 + \alpha \pmod b$
6: $\alpha = (r_0 + \alpha)/b$
7: $R(t) = R(t) - \beta \cdot \Theta(t) \pmod q$
8: $R(t) = R(t) - (r_0 - \beta) \pmod q$
9: $R(t) = R(t) \odot \Gamma(t) \pmod q$
10: **end for**
11: **return** $R(t)$

We will now describe the spectral point multiplication method for elliptic curves. We would like to remark that different representations of points on elliptic curves brings various realizations of the elliptic curve system which are suitable for different purposes. The two common representations are deduced by presenting the curve in the *affine* and *projective* coordinates. The former gives a straightforward representation involving inversions in the finite field, while the latter replaces inversions by multiplications. In general, this is desired since the inversion operation in $GF(n)$ is more time- and resource-intensive operation than the multiplication.

8.2 Spectral Projective Point Multiplication (SPPM)

We describe the spectral point multiplication method for projective coordinates. The SPPM method computes $\mathbb{Q} = e \times \mathbb{P}$ given the integer e and the point $\mathbb{P}$. The underlying field is $GF(n)$, and therefore, we need to setup a mod n spectral arithmetic as was the case for the modular exponentiation operation. The preprocessing step of the SPPM method is essentially the same the preprocessing step of the SMM method (Preprocessing with n): Given n, we need to compute ν_0, θ, $\Theta(t)$, λ, δ, $\Delta(t)$, and and $K(t)$. After these computations, we start the Preprocessing with $\mathbb{P}$ phase, and move into the Exponentiation Loop and Postprocessing phases.

Preprocessing with $\mathbb{P}$: Given $\mathbb{P} = (x, y, z)$, obtain $\mathbb{P}(t) = (x(t), y(t), z(t))$.

1. Compute the point $\mathbb{P}'(t) = (X(t), Y(t), Z(t)) = (\mathrm{DFT}[x(t)], \mathrm{DFT}[y(t)], \mathrm{DFT}[z(t)])$.
2. Assign $\mathbb{Q}'(t) = \mathbb{O}'(t) = (0, K(t), 0)$. Note that $\mathbb{O} = (0, 1, 0)$ is the projective coordinate representation of the point at infinity and $\mathbb{O}'(t) = (0, K(t), 0)$ is its DFT.
3. Use the modified Spectral Modular Multiplication (SMM2) method to compute

$$\overline{\mathbb{P}}'(t) = (\overline{X}(t), \overline{Y}(t), \overline{Y}(t))$$

as follows

$$\begin{aligned} \overline{X}(t) &= \mathrm{SMM2}[X(t), \Delta(t)] \ , \\ \overline{Y}(t) &= \mathrm{SMM2}[Y(t), \Delta(t)] \ , \\ \overline{Z}(t) &= \mathrm{SMM2}[Z(t), \Delta(t)] \ . \end{aligned}$$

4. Use the SMM2 method to compute

$$\overline{\mathbb{Q}}'(t) = (0, \overline{K}(t), 0)$$

such that

$$\overline{K}(t) = \text{SMM2}[K(t), \Delta(t)] \ .$$

Exponentiation Loop: The exponentiation operation is performed as soon as the j-bit exponent e is available. Let the binary expansion of e be $(e_{j-1}e_{j-2}\cdots e_1e_0)_2$. The exponentiation operation needs $\overline{\mathbb{Q}}'(t)$ and $\overline{\mathbb{P}}'(t)$ in addition to the exponent e. The exponentiation method (the point multiplication) method relies on elliptic curve point doubling and point addition methods. Since we work in the spectral domain and in the projective coordinate systems, we name these methods as the Spectral Projective Point Doubling (SPPD) and the Spectral Projective Point Additions (SPPA) methods.

for $i = j-1$ **downto** 0
  $\overline{\mathbb{Q}}'(t) = \text{SPPD}[\overline{\mathbb{Q}}'(t)]$
  if $e_i = 1$ **then** $\overline{\mathbb{Q}}'(t) = \text{SPPA}[\overline{\mathbb{Q}}'(t), \overline{\mathbb{P}}'(t)]$

Postprocessing: After the additive exponentiation loop is completed, we will have a final value of

$$\overline{\mathbb{Q}}'(t) = (\overline{X}(t), \overline{Y}(t), \overline{Z}(t)) \ .$$

This vector now needs to be brought back to the time domain.

1. Obtain $\mathbb{Q}'(t) = (X(t), Y(t), Z(t))$ using the SMM2 method by multiplying $K(t)$ as

$$\begin{aligned} X(t) &= \text{SMM2}[\overline{X}(t), K(t)] \ , \\ Y(t) &= \text{SMM2}[\overline{Y}(t), K(t)] \ , \\ Z(t) &= \text{SMM3}[\overline{Z}(t), K(t)] \ . \end{aligned}$$

2. Obtain $\mathbb{Q}(t) = (x(t), y(t), z(t))$ using the Inverse DFT function as follows

$$\begin{aligned} x(t) &= \text{IDFT}[X(t)] \ , \\ y(t) &= \text{IDFT}[Y(t)] \ , \\ z(t) &= \text{IDFT}[Z(t)] \ . \end{aligned}$$

Output: The point $\mathbb{Q}(t) = (x(t), y(t), z(t))$ is the output of the SPPM method, such that

$$\mathbb{Q}(t) = e \times \mathbb{P}(t) \ .$$

8.3 Spectral Projective Point Addition (SPPA)

Let $\mathbb{P}_0(t) = (X_0(t), Y_0(t), Z_0(t))$ and $\mathbb{P}_1(t) = (X_1(t), Y_1(t), Z_1(t))$ be spectral representation of two points on an elliptic curve E. The SPPA algorithm computes the projective point addition $\mathbb{P}_2 = \mathbb{P}_0 + \mathbb{P}_1$ in the spectral domain. We will denote the operation using

$$\mathbb{P}_2(t) = (X_2(t), Y_2(t), Z_2(t)) = \text{SPPA}[\mathbb{P}_0(t), \mathbb{P}_1(t)] \ .$$

The steps of the SPPA method are given below.

$$\begin{aligned} \Psi_0(t) &= \text{SMM2}[X_0(t), \text{SMM2}[Z_1(t), Z_1(t)]] \ , \\ \Psi_1(t) &= \text{SMM2}[\text{SMM2}[Y_0(t), Z_1(t)], \text{SMM2}[Z_1(t), Z_1(t)]] \ , \end{aligned}$$

$$
\begin{aligned}
\Psi_2(t) &= \mathrm{SMM2}[X_1(t), \mathrm{SMM2}[Z_0(t), Z_0(t)]] \ , \\
\Psi_3(t) &= \mathrm{SMM2}[\mathrm{SMM2}[Y_1(t), Z_0(t)], \mathrm{SMM2}[Z_0(t), Z_0(t)]] \ , \\
\Psi_4(t) &= \Psi_0(t) - \Psi_2(t) \ , \\
\Psi_5(t) &= \Psi_0(t) + \Psi_2(t) \ , \\
\Psi_6(t) &= \Psi_1(t) - \Psi_3(t) \ , \\
\Psi_7(t) &= \Psi_1(t) + \Psi_3(t) \ , \\
Z_2(t) &= \mathrm{SMM2}[Z_0(t), \mathrm{SMM2}[Z_1(t), \Psi_4(t)]] \ , \\
X_2(t) &= \mathrm{SMM2}[\Psi_6(t), \Psi_6(t)] - \mathrm{SMM2}[\Psi_5(t), \mathrm{SMM2}[\Psi_4(t), \Psi_4(t)]] \ , \\
\Psi_8(t) &= \mathrm{SMM2}[\Psi_5(t), \mathrm{SMM2}[\Psi_4(t), \Psi_4(t)]] - 2X_2(t) \ , \\
2Y_2(t) &= \mathrm{SMM2}[\Psi_8(t), \Psi_6(t)] - \mathrm{SMM2}[\mathrm{SMM2}[\Psi_7(t), \Psi_4(t)], \mathrm{SMM2}[\Psi_4(t), \Psi_4(t)]] \ .
\end{aligned}
$$

8.4 Spectral Projective Point Doubling (SPPD)

The SPPD algorithm computes the projective doubling $2 \times \mathbb{P}_1(t)$ operation. We will denote the operation using

$$\mathbb{P}_2(t) = (X_2(t), Y_2(t), Z_2(t)) = 2 \times \mathbb{P}_1(t) = \mathrm{SPPD}[\mathbb{P}_1(t)] \ .$$

The steps are given below.

$$
\begin{aligned}
\Psi_0(t) &= 3 \cdot \mathrm{SMM2}[X_1(t), X_1(t)] + a \cdot \mathrm{SMM2}[\mathrm{SMM2}[Z_1(t), Z_1(t)], \mathrm{SMM2}[Z_1(t), Z_1(t)]] \ , \\
Z_2(t) &= 2 \cdot \mathrm{SMM2}[Y_1(t), Z_1(t)] \ , \\
\Psi_1(t) &= 4 \cdot \mathrm{SMM2}[X_1(t), \mathrm{SMM2}[Y_1(t), Y_1(t)]] \ , \\
X_2(t) &= \mathrm{SMM2}[\Psi_0(t), \Psi_0(t)] - 2\Psi_1(t) \ , \\
\Psi_2(t) &= 8 \cdot \mathrm{SMM2}[\mathrm{SMM2}[Y_1(t), Y_1(t)], \mathrm{SMM2}[Y_1(t), Y_1(t)]] \ , \\
Y_2(t) &= \mathrm{SMM2}[\Psi_0(t), \Psi_0(t) - X_2(t)] - \Psi_2(t) \ .
\end{aligned}
$$

8.5 Spectral Affine Point Multiplication (SAPM)

If the affine coordinates used to represent the curve, the addition and doubling formulae get simpler but one needs to deal with the inversions in the finite field. The flow the the point multiplication algorithm is same as the projective case with a fewer coordinates. The preprocessing step of the SAPM method is exactly the same the preprocessing step of the SPPM method: Given n, we need to compute ν_0, θ, $\Theta(t)$, λ, δ, $\Delta(t)$, and and $K(t)$. After these computations, we start the Preprocessing with $\mathbb{P}$ phase, and move into the Exponentiation Loop and Postprocessing phases.

Preprocessing with $\mathbb{P} = (x, y)$: Given $\mathbb{P}$, obtain $\mathbb{P}(t) = (x(t), y(t))$.

1. Compute the point $\mathbb{P}'(t) = (X(t), Y(t)) = (\mathrm{DFT}[x(t)], \mathrm{DFT}[y(t)])$.
2. Assign $\mathbb{Q}'(t) = \mathbb{O}'(t) = (0, K(t))$. Note that $\mathbb{O} = (0, 1)$ is the projective coordinate representation of the point at infinity and $\mathbb{O}'(t) = (0, K(t))$ is its DFT.
3. Use the SMM2 method to compute $\overline{\mathbb{P}}(t) = (\overline{X}(t), \overline{Y}(t))$, where

$$
\begin{aligned}
\overline{X}(t) &= \mathrm{SMM2}[X(t), \Delta(t)] \ , \\
\overline{Y}(t) &= \mathrm{SMM2}[Y(t), \Delta(t)] \ .
\end{aligned}
$$

4. Use the SMM2 method to compute

$$\overline{\mathbb{Q}}'(t) = (\overline{K}(t), 0)$$

such that

$$\overline{K}(t) = \text{SMM2}[K(t), \Delta(t)] \ .$$

Exponentiation Loop: The exponentiation operation is performed as soon as the j-bit exponent e is available. Let the binary expansion of e be $(e_{j-1}e_{j-2}\cdots e_1e_0)_2$. The exponentiation operation needs $\overline{\mathbb{Q}}'(t)$ and $\overline{\mathbb{P}}'(t)$ as input in addition to the exponent e.

for $i = j-1$ **downto** 0
  $\overline{\mathbb{Q}}'(t) = \text{SAPD}[\overline{\mathbb{Q}}'(t)]$
  if $e_i = 1$ **then** $\overline{\mathbb{Q}}'(t) = \text{SAPA}[\overline{\mathbb{Q}}'(t), \overline{\mathbb{P}}'(t)]$

Postprocessing: After the additive exponentiation loop is completed, we will compute the final value of $\overline{\mathbb{Q}}'(t) = (\overline{X}(t), \overline{Y}(t))$. This vector will now be brought back to the time domain as follows.

1. Obtain $\mathbb{Q}'(t) = (X(t), Y(t))$ using the SMM2 method by multiplying $K(t)$ as

$$\begin{aligned} X(t) &= \text{SMM2}[\overline{X}(t), K(t)] \ , \\ Y(t) &= \text{SMM2}[\overline{Y}(t), K(t)] \ . \end{aligned}$$

2. Obtain $\mathbb{Q}(t) = (x(t), y(t))$ using the Inverse DFT function as follows

$$\begin{aligned} x(t) &= \text{IDFT}[X(t)] \ , \\ y(t) &= \text{IDFT}[Y(t)] \ . \end{aligned}$$

Output: The point $\mathbb{Q}(t) = (x(t), y(t))$ is the output of the SPPM method, such that

$$\mathbb{Q}(t) = e \times \mathbb{P}(t) \ .$$

8.5.1 Spectral Affine Point Addition (SAPA)

Let $\mathbb{P}_0(t) = (X_0(t), Y_0(t))$ and $\mathbb{P}_1(t) = (X_1(t), Y_1(t))$ be spectral representation of two points on an elliptic curve E. The SAPA algorithm computes the affine point addition $\mathbb{P}_2 = \mathbb{P}_0 + \mathbb{P}_1$ in the spectral domain. We will denote the operation using

$$\mathbb{P}_2(t) = (X_2(t), Y_2(t)) = \text{SAPA}[\mathbb{P}_0(t), \mathbb{P}_1(t)] \ .$$

The steps of the SAPA method are given below;

$$\begin{aligned} \Psi(t) &= \text{SMM2}[Y_1(t) - Y_0(t), (X_1(t) - X_0(t))^{-1}] \ , \\ X_2(t) &= \text{SMM2}[\Psi(t), \Psi(t)] - X_1(t) - X_0(t) \ , \\ Y_2(t) &= \text{SMM2}[X_0(t) - X_2(t), \Psi(t)] - X_2(t) - Y_0(t) \ . \end{aligned}$$

8.6 Spectral Affine Point Doubling (SAPD)

The SAPD algorithm computes the affine point doubling $2 \times \mathbb{P}_1(t)$ operation. We will denote the operation using

$$\mathbb{P}_2(t) = (X_2(t), Y_2(t)) = 2 \times \mathbb{P}_1(t) = \text{SAPD}[\mathbb{P}_1(t)] \ .$$

The steps are given below.

$$\begin{aligned} \Psi(t) &= \text{SMM2}[3X_0(t) + a(t), (2Y_1(t))^{-1}] \ , \\ X_2(t) &= \text{SMM2}[\Psi(t), \Psi(t)] - 2X_1(t) \ , \\ Y_2(t) &= \text{SMM2}[X_0(t) - X_2(t), \Psi(t)] - X_2(t) - Y_0(t) \ . \end{aligned}$$

8.7 Parameter Selection for Elliptic Curve Cryptography

In Section 6, we described the parameter selection methodology for the SME method enriched by some sample parameters giving the key sizes around the most popular key sizes. In this section, we will present some similar examples for the ECC. Practically, the SMM2 and SMM algorithms give the same bounds enforced by the inequalities (1) and (2). The improvements desribed in Section 6.1 are also applicable to SMM2 method. In **Table 7**, we demonstrate the suitable parameters for the SPM (valid for both PSPM and ASPM). We would like to add that the main characteristic of the ECC is having shorter key sizes ranges from 160 bits to 540 bits. Thus, these tables can be seen as a continuation of the parameter selection tables of Section 6.

Table 7: Standard and improved parameter selections for ECC.

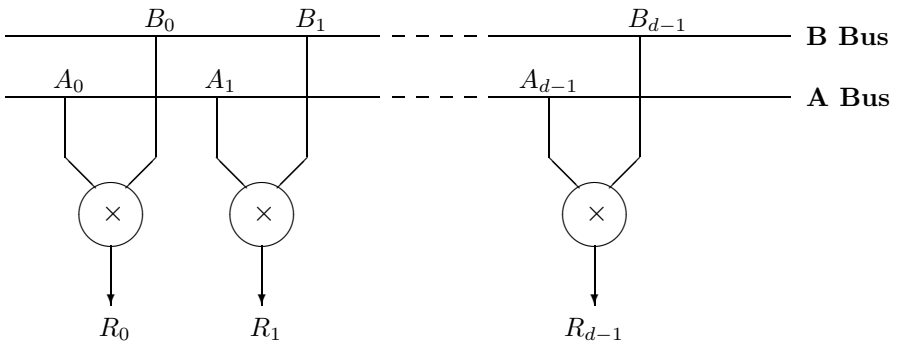
Bits k	Ring q	DFT d	Root w	Wordsize u	Words s
176	$2^{43}-1$	43	2	8	22
185	$2^{37}-1$	74	-2	5	37
190	$(2^{38}-1)/3$	76	-2	5	38
192	$2^{47}-1$	47	2	8	24
234	$(2^{51}-1)/7$	51	2	9	26
270	$2^{53}-1$	53	2	10	27
384	$2^{64}+1$	64	4	12	32
580	$(2^{58}-1)/3$	116	-2	10	58
171	$(2^{37}+1)/3$	37	4	9	19
186	$2^{31}-1$	62	-2	6	31
190	$2^{37}-1$	37	2	10	19
210	$(2^{41}+1)/3$	41	4	10	21
224	$2^{32}+1$	64	2	7	32
231	$2^{41}-1$	41	2	11	21
264	$2^{43}-1$	43	2	12	22
296	$2^{37}-1$	74	-2	8	37
405	$(2^{53}+1)/3$	53	4	15	27
410	$2^{41}-1$	82	-2	10	41
540	$(2^{59}+1)/3$	59	2	19	30
564	$2^{47}-1$	94	-2	12	47

9 Hardware Architectures for Spectral Modular Arithmetic

The core part of the SME, SPPM, SAPM methods consists of the SMM and SMM2 algorithms. These two multiplication algorithms (SMM and SMM2) are same except the representation set of Z_q. From a design point of view, this difference is quite insignificant, and can be dealt with in the circuit level. In this section, we will describe the hardware architecture implementing the SMM method by going through its steps, as described in Section 3.

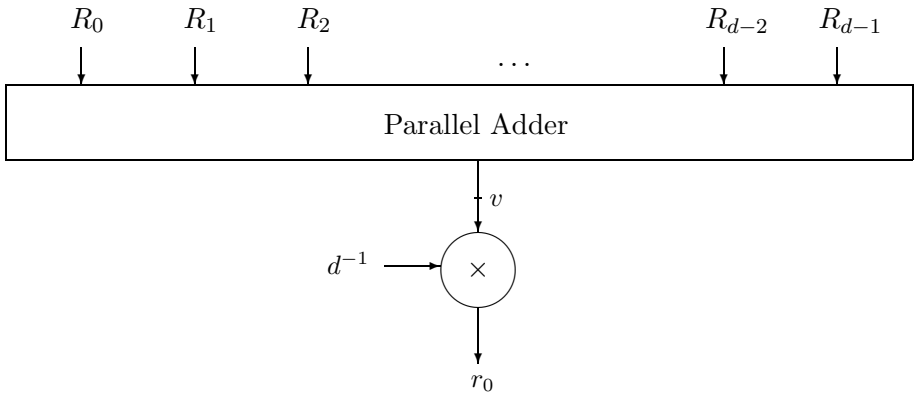
- In Step 1, the convolution property is employed. Given the vectors $A(t)$ and $B(t)$, we compute $R(t)$ such that $R_i = A_i B_i \pmod q$ for $i = 0, 1, \ldots, d-1$. Recall that our targeted rings are the Mersenne or Fermat rings for which q is of the form $2^v \pm 1$. Hence, these multiplications can be realized by employing $v \times v$ modulo multipiers and the complexities of these multipliers are not difficult from the usual integer multiplication. This computation is accomplished using the hardware architecture given in **Figure 1**.

Figure 1: The archictecture for Step 1 of SMM.



- For simplicity, the reduction steps which correspond to the loop with i is divided into two parts:
  - **Parameter Generation:** This corresponds to Steps 4, 5 and 6. Here, we compute the parameters r_0, α, and β, and feed feed them to the main processing units.
  - **Processing Engine:** This corresponds to steps 7, 8, and 9, in which we add a multiple of the modulus to the partial sum and then divide it by the base.
- In **Parameter Generation**, Step 4 corresponds to a partial interpolation in which a d-input multi-operand addition followed by a multiplication by d^{-1} is performed in order to find the zeroth coefficient of the polynomial. **Figure 2** shows the architecture for these computations.

Figure 2: Partial interpolation.

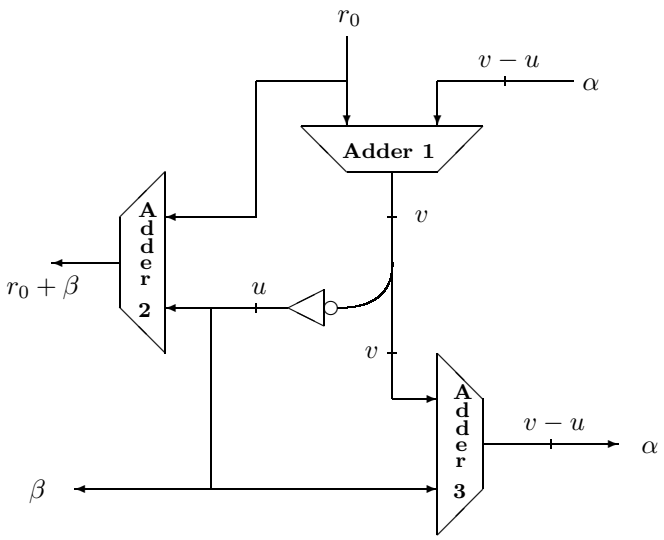


Observe that d^{-1} is a constant v-bit number (recall that $q = 2^v \pm 1$), therefore, this can be accomplished using a multi-operand addition.

Whenever d and w are both a power of 2, multiplication by d^{-1} can be replaced with shifts. This can be seen as follows: Let $w = 2^l$, and thus, we have $w^d = 2^{ld} = 1 \bmod q$. We can write d^{-1} as $d^{-1} = 2^{ld - \log d} \bmod q$, hence, multiplication by d^{-1} modulo q can be accomplished with a $ld - \log d$ bit circular shift. Therefore, for special Fermat or Mersenne rings, it is possible drop the multiplication by d^{-1} in **Figure 2**.

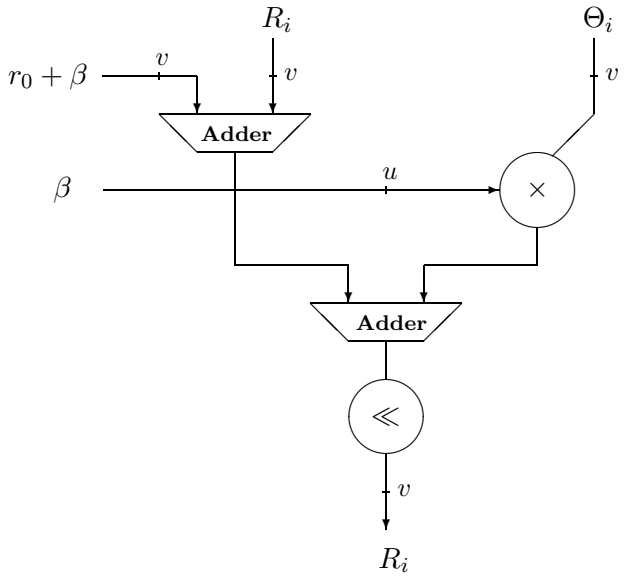
- Steps 5 and 6 as seen in the **Figure 3** are called the *Parameter Generation Logic (PGL)* which computes the parameters $r_0 + \beta$ and β. The adders seen in **Figure 3** are the usual v-bit adders and they do not need modular reductions since $\alpha, (r_0 + \alpha), (r_0 + \alpha + \beta) < q$.

Figure 3: Parameter generation logic.



- The **Processing Engine** is the most resource-consuming stage and it corresponds to Steps 7, 8, and 9. In **Figure 4**, we give the architecture of a single processing unit. The processing engine consists of d such units.

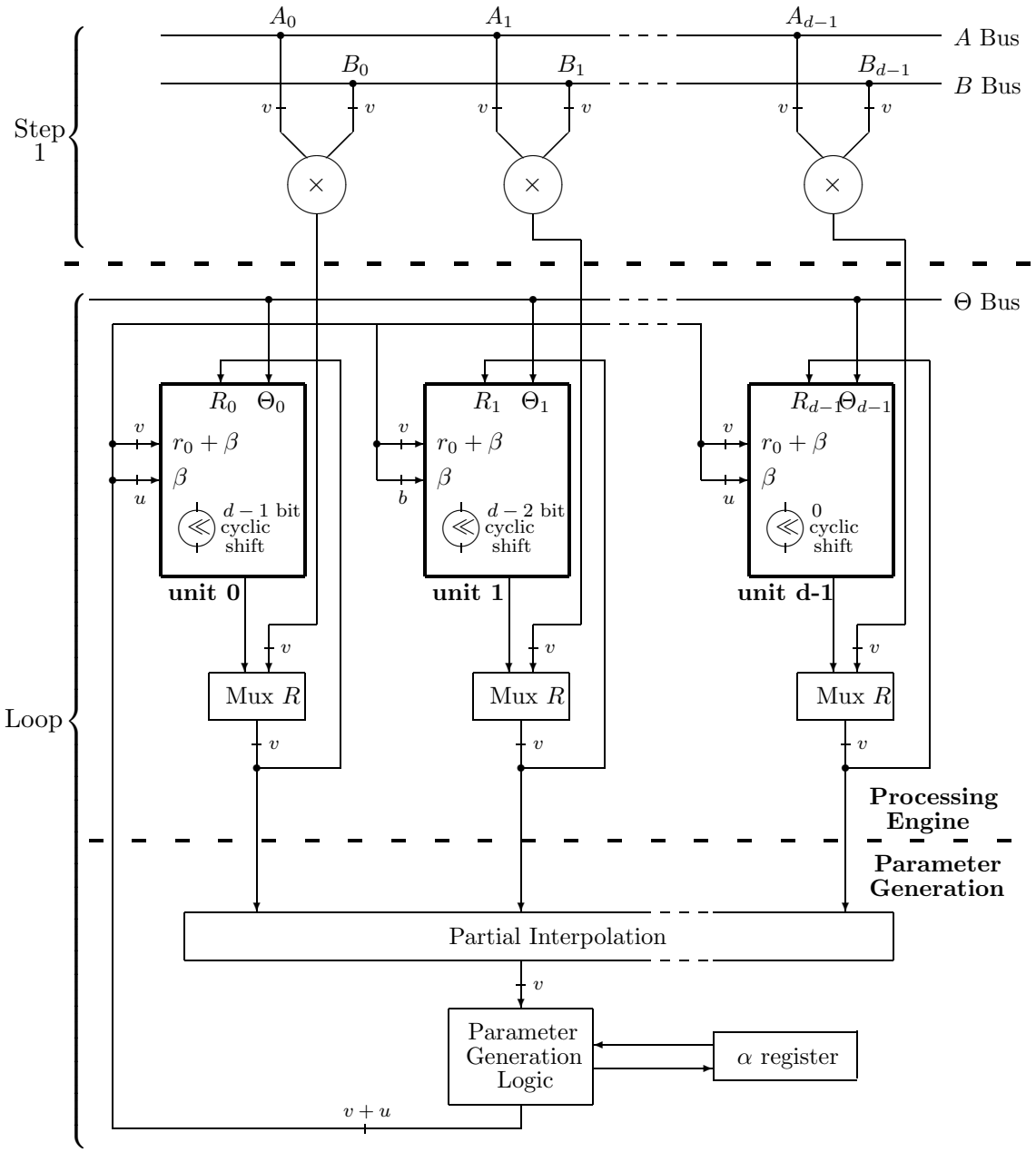
Figure 4: A processing unit.



Both adders in **Figure 4** are modulo q adders. The shift operation at the bottom of the figure corresponds to Step 9 of the SMM core. As we pick w as a power of 2, the multiplications with the coeeficients of $\Gamma(t)$ correspond to the constant $d-1-i$ bit shifts for processing units R_i where $i = 1, 2, \ldots, d-1$.

- We obtain the entire architecture of the SMM method by combining these individual pieces. The architecture is illustrated in **Figure 5**. In this architecture, the outputs of the convolution step (Step 1) feed the R MUXes. For the initial case, each MUX R chooses the input from Step 1, and then the reduction loop starts. The loop runs d times: at every run, the outputs of the processing units are passed to the interpolation and also fed back to the unit itself. The processing engine waits until the parameters $r_0 + \beta$ and β are generated from the parameter generation logic. After d runs of the loop, the processing units from 0 to $d-1$ outputs the coefficient of the resultant spectral polynomial $R(t)$. It is important to realize that in this architecture:
  - All processing units work in parallel.
  - The units are not completely identical. The cyclic shifts are different for each unit.
  - We do not need a cyclic shifter in unit $d-1$
  - The same $r_0 + \beta$ and β are passed to all of the processing units.

Figure 5: The hardware architecture of the SMM algorithm.



10 Appendix

Table 8: Suitable Fermat rings and the w and d values.

Ring	Prime Factors	w	d	w	d
$2^{16}+1$	65537	4	16	2	32
$2^{20}+1$	$17 \cdot 61681$	32	8	4100	16
$2^{24}+1$	$97 \cdot 257 \cdot 673$	8	16	$\sqrt{8}$	32
$2^{32}+1$	$641 \cdot 6700417$	4	32	2	64
$2^{40}+1$	$257 \cdot 4278255361$	32	16	$\sqrt{32}$	256
$2^{64}+1$	$274177 \cdot 67280421310721$	4	64	2	128
$2^{80}+1$	$414721 \cdot 44479210368001$	32	32	$\sqrt{32}$	1024
$2^{96}+1$	$641 \cdot 6700417 \cdot 18446744069414584321$	8	64	$\sqrt{8}$	128
$2^{112}+1$	$449 \cdot 2689 \cdot 65537 \cdot 183076097 \cdot 358429848460993$	2^7	32	$\sqrt{128}$	64
$2^{128}+1$	$59649589127497217 \cdot 5704689200685129054721$	4	128	2	256

Table 9: Suitable Mersenne rings and the w and d values.

Ring	Prime Factors	w	d	w	d
$2^{17}-1$	131071	2	17	-2	34
$2^{19}-1$	524287	2	19	-2	38
$2^{23}-1$	$47 \cdot 178481$	2	23	-2	46
$2^{29}-1$	$233 \cdot 1103 \cdot 2089$	2	29	-2	58
$2^{31}-1$	2147483647	2	31	-2	62
$2^{37}-1$	$223 \cdot 616318177$	2	37	-2	74
$2^{41}-1$	$13367 \cdot 164511353$	2	41	-2	82
$2^{43}-1$	$431 \cdot 9719 \cdot 2099863$	2	43	-2	86
$2^{47}-1$	$2351 \cdot 4513 \cdot 13264529$	2	47	-2	94
$2^{53}-1$	$6361 \cdot 69431 \cdot 20394401$	2	53	-2	106
$2^{59}-1$	$179951 \cdot 3203431780337$	2	59	-2	118
$2^{61}-1$	2305843009213693951	2	61	-2	122
$2^{67}-1$	$193707721 \cdot 761838257287$	2	67	-2	134
$2^{71}-1$	$228479 \cdot 48544121 \cdot 212885833$	2	71	-2	142
$2^{73}-1$	$439 \cdot 2298041 \cdot 9361973132609$	2	73	-2	146
$2^{79}-1$	$2687 \cdot 202029703 \cdot 1113491139767$	2	79	-2	158
$2^{83}-1$	$167 \cdot 57912614113275649087721$	2	83	-2	166
$2^{89}-1$	618970019642690137449562111	2	89	-2	178
$2^{97}-1$	$11447 \cdot 13842607235828485645766393$	2	97	-2	196
$2^{101}-1$	$7432339208719 \cdot 341117531003194129$	2	101	-2	202
$2^{103}-1$	$2550183799 \cdot 3976656429941438590393$	2	103	-2	206
$2^{107}-1$	162259276829213363391578010288127	2	107	-2	214
$2^{109}-1$	$745988807 \cdot 870035986098720987332873$	2	109	-2	218
$2^{113}-1$	$3391 \cdot 23279 \cdot 65993 \cdot 1868569 \cdot 1066818132868207$	2	113	-2	226
$2^{127}-1$	170141183460469231731687303715884105727	2	127	-2	254

Table 10A: Suitable pseudo Fermat rings and the w and d values.

Ring	Prime Factors	Modulus	w	d	w	d
$2^{17}+1$	$3 \cdot 43691$	$(2^{17}+1)/3$	$-2, 4$	17	2	34
$2^{19}+1$	$3 \cdot 174763$	$(2^{19}+1)/3$	$-2, 4$	19	2	38
$2^{20}+1$	$17 \cdot 61681$	$(2^{20}+1)/17$	4	20	2	40
$2^{21}+1$	$3^2 \cdot 43 \cdot 5419$	$(2^{21}+1)/9$	$-2, 4$	21	2	42
$2^{22}+1$	$5 \cdot 397 \cdot 2113$	$(2^{22}+1)/5$	4	22	2	44
$2^{23}+1$	$3 \cdot 2796203$	$(2^{23}+1)/3$	$-2, 4$	23	2	46
$2^{28}+1$	$17 \cdot 15790321$	$(2^{28}+1)/17$	4	28	2	56
$2^{29}+1$	$3 \cdot 59 \cdot 3033169$	$(2^{29}+1)/3$	$-2, 4$	29	2	58
$2^{31}+1$	$3 \cdot 715827883$	$(2^{31}+1)/3$	$-2, 4$	31	2	62
$2^{34}+1$	$5 \cdot 137 \cdot 953 \cdot 26317$	$(2^{34}+1)/5$	4	34	2	68
$2^{37}+1$	$3 \cdot 25781083 \cdot 1777$	$(2^{37}+1)/3$	$-2, 4$	37	2	74
$2^{38}+1$	$5 \cdot 229 \cdot 457 \cdot 525313$	$(2^{38}+1)/5$	4	38	2	76
$2^{39}+1$	$3^2 \cdot 22366891 \cdot 2731$	$(2^{39}+1)/9$	$-2, 4$	39	2	78
$2^{40}+1$	$257 \cdot 4278255361$	$(2^{40}+1)/257$	4	40	2	80
$2^{41}+1$	$3 \cdot 83 \cdot 8831418697$	$(2^{41}+1)/3$	$-2, 4$	41	2	82
$2^{43}+1$	$3 \cdot 2932031007403$	$(2^{43}+1)/3$	$-2, 4$	43	2	86
$2^{44}+1$	$17 \cdot 353 \cdot 2931542417$	$(2^{44}+1)/17$	4	44	2	88
$2^{46}+1$	$5 \cdot 277 \cdot 1013 \cdot 1657 \cdot 30269$	$(2^{46}+1)/5$	4	46	2	92
$2^{47}+1$	$3 \cdot 283 \cdot 165768537521$	$(2^{47}+1)/3$	$-2, 4$	47	2	94
$2^{52}+1$	$17 \cdot 308761441 \cdot 858001$	$(2^{52}+1)/17$	4	52	2	102
$2^{53}+1$	$3 \cdot 107 \cdot 28059810762433$	$(2^{53}+1)/3$	$-2, 4$	53	2	106
$2^{56}+1$	$257 \cdot 54410972897 \cdot 5153$	$(2^{56}+1)/257$	4	56	2	112
$2^{57}+1$	$3^2 \cdot 571 \cdot 160465489 \cdot 174763$	$(2^{57}+1)/9$	$-2, 4$	57	2	114
$2^{58}+1$	$5 \cdot 107367629 \cdot 536903681$	$(2^{58}+1)/5$	4	58	2	116
$2^{59}+1$	$3 \cdot 1824726041 \cdot 37171 \cdot 2833$	$(2^{59}+1)/3$	$-2, 4$	59	2	118
$2^{60}+1$	$17 \cdot 241 \cdot 4562284561 \cdot 61681$	$(2^{60}+1)/17$	4	60	2	120
$2^{61}+1$	$3 \cdot 768614336404564651$	$(2^{61}+1)/3$	$-2, 4$	61	2	122
$2^{62}+1$	$5 \cdot 384773 \cdot 49477 \cdot 8681 \cdot 5581$	$(2^{62}+1)/5$	4	62	2	124
$2^{65}+1$	$3 \cdot 11 \cdot 131 \cdot 409891 \cdot 7623851 \cdot 2731$	$(2^{65}+1)/3$	$-2, 4$	65	2	130

Table 10B: Suitable pseudo Fermat rings and the w and d values.

Ring	Prime Factors	Modulus	w	d	w	d
$2^{66}+1$	$5 \cdot 13 \cdot 397 \cdot 4327489 \cdot 312709 \cdot 2113$	$(2^{66}+1)/5$	4	66	2	132
$2^{67}+1$	$3 \cdot 6713103182899 \cdot 7327657$	$(2^{67}+1)/3$	$-2, 4$	67	2	134
$2^{68}+1$	$17^2 \cdot 2879347902817 \cdot 354689$	$(2^{68}+1)/17^2$	4	68	2	136
$2^{71}+1$	$3 \cdot 56409643 \cdot 13952598148481$	$(2^{71}+1)/3$	$-2, 4$	71	2	142
$2^{73}+1$	$3 \cdot 1795918038741070627 \cdot 1753$	$(2^{73}+1)/3$	$-2, 4$	73	2	146
$2^{74}+1$	$5 \cdot 149 \cdot 593 \cdot 184481113 \cdot 231769777$	$(2^{74}+1)/5$	4	74	2	148
$2^{76}+1$	$17 \cdot 1217 \cdot 24517014940753 \cdot 148961$	$(2^{76}+1)/17$	4	76	2	152
$2^{79}+1$	$3 \cdot 201487636602438195784363$	$(2^{79}+1)/3$	$-2, 4$	79	2	158
$2^{82}+1$	$5 \cdot 181549 \cdot 12112549 \cdot 43249589 \cdot 10169$	$(2^{82}+1)/5$	4	82	2	164
$2^{83}+1$	$3 \cdot 499 \cdot 1163 \cdot 13455809771 \cdot 155377 \cdot 2657$	$(2^{83}+1)/3$	$-2, 4$	83	2	166
$2^{85}+1$	$3 \cdot 11 \cdot 26831423036065352611 \cdot 43691$	$(2^{85}+1)/33$	$-2, 4$	85	2	170
$2^{86}+1$	$5 \cdot 173 \cdot 1759217765581 \cdot 500177 \cdot 101653$	$(2^{86}+1)/5$	4	86	2	172
$2^{87}+1$	$3^2 \cdot 59 \cdot 96076791871613611 \cdot 3033169$	$(2^{87}+1)/531$	$-2, 4$	87	2	174
$2^{88}+1$	$257 \cdot 43872038849 \cdot 119782433 \cdot 229153$	$(2^{88}+1)/257$	4	88	2	176
$2^{89}+1$	$3 \cdot 179 \cdot 18584774046020617 \cdot 62020897$	$(2^{89}+1)/3$	$-2, 4$	89	2	178
$2^{91}+1$	$3 \cdot 43 \cdot 25829691707 \cdot 1210483 \cdot 2731 \cdot 224771$	$(2^{91}+1)/129$	$-2, 4$	91	2	182
$2^{92}+1$	$17 \cdot 291280009243618888211558641$	$(2^{92}+1)/17$	4	92	2	184
$2^{93}+1$	$3^2 \cdot 529510939 \cdot 2903110321 \cdot 715827883$	$(2^{93}+1)/9$	$-2, 4$	93	2	186
$2^{94}+1$	$5 \cdot 7484047069 \cdot 140737471578113 \cdot 3761$	$(2^{94}+1)/5$	4	94	2	188
$2^{96}+1$	$641 \cdot 18446744069414584321 \cdot 6700417$	$(2^{96}+1)/641$	4	96	2	192
$2^{97}+1$	$3 \cdot 971 \cdot 1553 \cdot 1100876018364883721 \cdot 31817$	$(2^{97}+1)/3$	$-2, 4$	97	2	194
$2^{101}+1$	$3 \cdot 845100400152152934331135470251$	$(2^{101}+1)/3$	$-2, 4$	101	2	202
$2^{103}+1$	$3 \cdot 8142767081771726171 \cdot 415141630193$	$(2^{103}+1)/3$	$-2, 4$	103	2	206
$2^{104}+1$	$257 \cdot 78919881726271091143763623681$	$(2^{104}+1)/257$	4	104	2	208
$2^{106}+1$	$5 \cdot 15358129 \cdot 586477649 \cdot 1801439824104653$	$(2^{106}+1)/5$	4	106	2	212
$2^{107}+1$	$3 \cdot 643 \cdot 84115747449047881488635567801$	$(2^{107}+1)/3$	$-2, 4$	107	2	214
$2^{109}+1$	$3 \cdot 207775684736234886312817 9 \cdot 104124649$	$(2^{109}+1)/3$	$-2, 4$	109	2	218

Table 10C: Suitable pseudo Fermat rings and the w and d values.

Ring	Prime Factors	Modulus	w	d	w	d
$2^{111}+1$	$3^2 \cdot 1777 \cdot 3331 \cdot 17539$ $\cdot 25781083 \cdot 107775231312019$	$(2^{111}+1)/9$	$-2, 4$	111	2	222
$2^{113}+1$	$3 \cdot 227 \cdot 48817$ $\cdot 636190001 \cdot 491003369344660409$	$(2^{113}+1)/3$	$-2, 4$	113	2	226
$2^{114}+1$	$5 \cdot 13 \cdot 229 \cdot 457 \cdot 131101$ $\cdot 160969 \cdot 525313 \cdot 275415303169$	$(2^{114}+1)/65$	$-2, 4$	114	2	228
$2^{116}+1$	$17 \cdot 59393$ $\cdot 822801951671441198323905681 77$	$(2^{116}+1)/17$	4	116	2	232
$2^{118}+1$	$5 \cdot 1181 \cdot 3541 \cdot 157649$ $\cdot 174877 \cdot 5521693 \cdot 104399276341$	$(2^{118}+1)/5$	$-2, 4$	118	2	236
$2^{120}+1$	$97 \cdot 257 \cdot 673 \cdot 394783681$ $\cdot 46908728641 \cdot 4278255361$	$(2^{120}+1)/257$	32	48	$\sqrt{32}$	96
$2^{121}+1$	$3 \cdot 683 \cdot 117371$ $\cdot 1105418458279780045573606 1107$	$(2^{121}+1)/4098$	$-2, 4$	121	2	242
$2^{122}+1$	$5 \cdot 733 \cdot 1709 \cdot 3456749$ $\cdot 8831418697 \cdot 13194317913029593$	$(2^{122}+1)/5$	4	122	2	244
$2^{123}+1$	$3^2 \cdot 83 \cdot 739 \cdot 165313$ $\cdot 8831418697 \cdot 13194317913029593$	$(2^{123}+1)/747$	$-2, 4$	123	2	146
$2^{124}+1$	$17 \cdot 290657 \cdot 3770202641$ $\cdot 1141629180401976895873$	$(2^{124}+1)/17$	4	124	2	248
$2^{127}+1$	$3 \cdot 5671372782015641057722910123862803 5243$	$(2^{127}+1)/3$	$-2, 4$	127	2	254

Table 11A: Suitable pseudo Mersenne rings and the w and d values.

Ring	Prime Factors	Modulus	w	d	w	d
$2^{25}-1$	$31 \cdot 601 \cdot 1801$	$(2^{25}-1)/31$	2	25	-2	50
$2^{26}-1$	$3 \cdot 2731 \cdot 8191$	$(2^{26}-1)/3$	2	26	-2	52
$2^{27}-1$	$7 \cdot 73 \cdot 262657$	$(2^{27}-1)/511$	2	27	-2	54
$2^{34}-1$	$3 \cdot 43691 \cdot 131071$	$(2^{34}-1)/3$	2	34	-2	68
$2^{35}-1$	$31 \cdot 71 \cdot 127 \cdot 122921$	$(2^{35}-1)/3937$	2	35	-2	70
$2^{38}-1$	$3 \cdot 174763 \cdot 524287$	$(2^{38}-1)/3$	2	38	-2	76
$2^{39}-1$	$7 \cdot 79 \cdot 8191 \cdot 121369$	$(2^{39}-1)/7$	2	39	-2	78
$2^{46}-1$	$3 \cdot 47 \cdot 178481 \cdot 2796203$	$(2^{46}-1)/3$	2	46	4	23
$2^{49}-1$	$127 \cdot 4432676798593$	$(2^{49}-1)/127$	2	49	-2	98
$2^{51}-1$	$7 \cdot 103 \cdot 2143 \cdot 11119 \cdot 131071$	$(2^{51}-1)/7$	2	51	-2	102
$2^{57}-1$	$7 \cdot 32377 \cdot 524287 \cdot 1212847$	$(2^{57}-1)/7$	2	57	-2	114
$2^{58}-1$	$3 \cdot 59 \cdot 233 \cdot 1103 \cdot 2089 \cdot 3033169$	$(2^{58}-1)/3$	2	58	4	29
$2^{62}-1$	$3 \cdot 715827883 \cdot 2147483647$	$(2^{62}-1)/3$	2	62	-2	124
$2^{64}-1$	$3 \cdot 5 \cdot 17 \cdot 257 \cdot 641 \cdot 65537 \cdot 6700417$	$(2^{64}-1)/255$	2	64	-2	128
$2^{65}-1$	$31 \cdot 8191 \cdot 145295143558111$	$(2^{65}-1)/31$	2	65	-2	130
$2^{74}-1$	$3 \cdot 223 \cdot 1777 \cdot 25781083 \cdot 616318177$	$(2^{74}-1)/3$	2	74	-2	148
$2^{75}-1$	$7 \cdot 31 \cdot 151 \cdot 601 \cdot 1801 \cdot 100801 \cdot 10567201$	$(2^{75}-1)/217$	2	75	-2	150
$2^{78}-1$	$3^2 \cdot 7 \cdot 79 \cdot 2731 \cdot 8191 \cdot 121369 \cdot 22366891$	$(2^{78}-1)/63$	2	78	4	39
$2^{82}-1$	$3 \cdot 83 \cdot 13367 \cdot 164511353 \cdot 8831418697$	$(2^{82}-1)/3$	2	82	4	41
$2^{85}-1$	$31 \cdot 131071 \cdot 9520972806333758431$	$(2^{85}-1)/31$	2	85	-2	170
$2^{86}-1$	$3 \cdot 431 \cdot 9719 \cdot 2099863 \cdot 2932031007403$	$(2^{86}-1)/3$	2	86	4	43
$2^{91}-1$	$127 \cdot 911 \cdot 8191 \cdot 112901153 \cdot 23140471537$	$(2^{91}-1)/127$	2	91	-2	182
$2^{93}-1$	$7 \cdot 2147483647 \cdot 658812288653553079$	$(2^{93}-1)/7$	2	93	-2	186
$2^{94}-1$	$3 \cdot 283 \cdot 2351 \cdot 4513 \cdot$ $13264529 \cdot 165768537521$	$(2^{94}-1)/3$	2	94	4	47
$2^{106}-1$	$3 \cdot 107 \cdot 6361 \cdot 69431 \cdot 20394401$ $\cdot 28059810762433$	$(2^{106}-1)/3$	2	106	4	53

Table 11B: Suitable pseudo Mersenne rings and the w and d values.

Ring	Prime Factors	Modulus	w	d	w	d
$2^{111}-1$	$7 \cdot 223 \cdot 321679 \cdot 26295457 \cdot 616318177$ $\cdot 319020217$	$(2^{111}-1)/7$	2	111	-2	222
$2^{114}-1$	$3^2 \cdot 7 \cdot 571 \cdot 32377 \cdot 174763 \cdot$ $524287 \cdot 1212847 \cdot 160465489$	$(2^{114}-1)/63$	2	114	4	57
$2^{115}-1$	$31 \cdot 47 \cdot 14951 \cdot 178481 \cdot 4036961$ $\cdot 2646507710984041$	$(2^{115}-1)/1457$	2	115	-2	230
$2^{118}-1$	$3 \cdot 2833 \cdot 37171 \cdot 179951 \cdot$ $1824726041 \cdot 3203431780337$	$(2^{118}-1)/3$	2	118	4	59
$2^{121}-1$	$23 \cdot 89 \cdot 727 \cdot$ $1786393878363164227858270210279$	$(2^{121}-1)/2047$	2	121	-2	242
$2^{122}-1$	$3 \cdot 768614336404564651 \cdot$ 2305843009213693951	$(2^{122}-1)/3$	2	122	-2	244
$2^{128}-1$	$3 \cdot 5 \cdot 17 \cdot 257 \cdot 641 \cdot 65537 \cdot$ $274177 \cdot 6700417 \cdot 67280421310721$	$(2^{128}-1)/255$	2	128	4	64

References

[1] Ç. K. Koç. High-Speed RSA Implementation. TR 201, RSA Laboratories, 73 pages, November 1994.

[2] Ç. K. Koç. RSA Hardware Implementation. TR 801, RSA Laboratories, 30 pages, April 1996.

[3] R. E. Blahut. *Fast Algorithms for Digital Signal Processing*, Chapter 6, Addison-Wesley Publishing Company, 1985.

[4] H. J. Naussbaumer. *Fast Fourier Transform and Convolution Algorithms*, Chapter 8, Springer, Berlin, Germany, 1982.

[5] J. M. Pollard. The fast Fourier transform in a finite field. *Mathematics of Computation*, vol. 25, pp. 365–374, 1971.