Chapter 7
Spectral Modular Arithmetic for Cryptography

Gökay Saldamlı and Çetin Kaya Koç

7.1 Introduction

Most public-key cryptosystems require resource-intensive arithmetic calculations in certain mathematical structures such as finite fields, groups, and rings. The efficient realizations of the these operations, including *modular multiplication, inversion, and exponentiation* are at the center of research activities in cryptographic engineering. Note that, being modular, these operations involve sequential reduction steps.

Spectral techniques for integer multiplication have been known for over a quarter of a century. Using the spectral integer multiplication of Schönhage and Strassen [1], large to extremely large sizes of numbers can be multiplied efficiently. Such computations are needed when computing π to millions of digits of precision, factoring, and also big prime search projects.

A naive way of utilizing the spectral techniques for modular multiplication starts with computing the multiplication using possibly Schönhage–Strassen and then a reduction in the time domain follows. This approach is preferable if the input length is large enough to meet the asymptotic crossover of Schönhage–Strassen, assuming the reduction has a constant cost. Additionally, if the naive method is used for operations involving consecutive multiplications, because of the costly forward and backward transformation computations, the asymptotic crossover of these operations would be similar to what a single modular multiplication has. Unfortunately, these crossovers are larger than the key sizes of most public-key cryptosystems; thus, in practice, the naive way is hardly used.

On the other hand, modular multiplication can be performed on the Fourier representations of integers. In such a representation, multiplications are readily available by the convolution property. Therefore, operations involving several modular multiplications can be computed efficiently.

Eczacıbaşı Embedded Design Center, e-mail: gokay.saldamli@eczacibasi.com.tr · City University of Istanbul & University of California Santa Barbara, e-mail: koc@cryptocode.net

In the following section, we start with introducing some preliminary notation and a formal definition of Discrete Fourier Transform (DFT) over $\mathbb{Z}_q$ (i.e., the ring of integers with multiplication and addition modulo a positive integer q). Based on this new terminology, we describe the main idea of spectral modular arithmetic including *spectral modular multiplication* (SMM) and *spectral modular exponentiation* (SME) in Section 7.3.

Section 7.4 describes methodologies for selecting the parameters for SME in order to apply the algorithm to public-key cryptography and Section 7.5 reveals how these methods can be extended to extension fields including binary and mid-size characteristic extensions. In fact, we present suitable spectrum for ECC realizations both over binary and mid-size characteristic extensions.

The chapter is closed with some final comments and discussions on the current and future research activities of the presented material.

7.2 Notation and Background

Spectral techniques are widely accepted and used in the field of digital signal processing, hence most of its existing notation and concept come from this theory. For many reasons, the signals and admissible operations on these signals of such a theory are quite different from that of a theory of computer arithmetic. For instance, when using FFT (or convolution property) for integer multiplication, first we partition the inputs into words. Note that any small perturbation in the resulting words would completely change the represented integer. On the other hand, approximations on the signal components without changing the main characteristics of the original signal are allowable in digital signal processing.

Therefore, we believe that we need a more clear notation that would permit us to have a better understanding of spectral methods and their applications to computer arithmetic-related problems. While doing this, we follow a polynomial representation instead of the standard sequence representation of digital signal processing. Such a presentation is necessary for our needs and, moreover, it states the different nature of the number-representing signals from a classical signal processing analysis.

7.2.1 Evaluation Polynomials

We start with building a new terminology that binds the polynomials over $\mathbb{Z}$ to their evaluations. A similar construction can be formulated for polynomials over the rings other than the ring of integers.

Definition 7.1. Let x and $b > 0$ be integers. If $x(t)$ is a polynomial in $\mathbb{Z}[t]$ such that $x(b) = x$, then we say $x(t)$ is an **evaluation polynomial of** x.

Remark 7.1. For ease of notation we denote the evaluation polynomials by a pair $(x,x(t))$. Sometimes we even simply use $x(t)$; the reader should be aware of polynomials in this text should always be considered with their evaluations.

Remark 7.2. Note that the positive integer b is called the **base** or **radix** in the literature. In order not to have any confusion with the frequent usage of the word "radix" for another instance in FFT theory, we prefer to use the word "base" for b.

Throughout this text we assume that b is a *fixed* positive integer and we denote the set of all evaluation polynomials over $\mathbb{Z}$ by $\mathscr{B}$. Observe that if b is fixed, there exists a natural one-to-one correspondence between $\mathscr{B}$ and $\mathbb{Z}[t]$ which is given by $(x,x(t)) \mapsto x(t)$,

In fact, the base b representation of an integer gives a special evaluation polynomial. We particularly specify these as follows:

Definition 7.2. Let

$$x(t) = x_0 + x_1 t + \ldots + x_{d-1} t^{d-1} \in \mathbb{Z}[t]$$

be an evaluation polynomial of an integer x for a fixed base b. If the coefficients of $x(t)$ satisfy $0 \leq x_i < b$ for all $i = 0,1,\ldots,d-1$, $x(t)$ is called the **base evaluation polynomial** or simply the **base polynomial** .

Example 7.1. A base $2^k, k > 0$ representation of an integer x $((x_0 x_1 \ldots x_d)$ with $0 \leq x_i < 2^k$ for $i = 0,1,\ldots,d-1)$ has the base polynomial $x(t) = x_0 + x_1 t + x_2 t^2 \ldots + x_{d-1} t^{d-1}$, where $y(t) = (x_0 + x_1 b) + 0 \cdot t + x_2 t^2 + \ldots + x_{d-1} t^{d-1}$ is one of its evaluation polynomials.

As seen in Example 7.1, the evaluation polynomial (or sequence) of an integer x is not unique. Indeed, the same integer has infinitely many different evaluation polynomials. But note that the base polynomials (i.e., base representations) are unique.

Proposition 7.1. *Let $\mathscr{B}$ denote the set of all evaluation polynomials; then $(\mathscr{B}, \oplus, \otimes)$ is a ring with the following operations;*

$$(x,x(t)) \oplus (y,y(t)) = (x+y, x(t)+y(t))$$

$$(x,x(t)) \otimes (y,y(t)) = (xy, x(t)y(t))$$

where $x(t), y(t) \in \mathbb{Z}[t]$ and $x,y \in \mathbb{Z}$.

Proof. Since base b is fixed and the structures on the components come from $\mathbb{Z}$ and $\mathbb{Z}[t]$, it is easily seen that $(\mathscr{B}, \oplus)$ is an abelian group and $(\mathscr{B}, \otimes)$ is closed. Therefore, all we need to show is that the evaluation map is well defined on the components. This is trivial because $x+y = x(b)+y(b)$, and the distribution property comes naturally from this observation. Thus, $(\mathscr{B}, \oplus, \otimes)$ is a ring with identity $(1_\oplus, 1_\otimes) = (0,1)$.

Proposition 7.2. *The map $\phi : \mathscr{B} \to \mathbb{Z}[t]$ sending $(x, x(t)) \mapsto x(t)$ is a ring isomorphism.*

Proof. Since $b > 0$ is fixed, the evaluation $x = x(b)$ is also fixed, which implies that there exists a natural subjective map from $\mathscr{B}$ to $\mathbb{Z}[t]$ sending $(x, x(t)) \mapsto x(t)$ with a zero kernel.

Definition 7.3. If $x(t)$ and $y(t)$ are evaluation polynomials for the same integer x, then we write $x(t) \sim y(t)$ and say $x(t)$ is related to $y(t)$.

Proposition 7.3. *$x(t) \sim y(t)$ is an equivalence relation.*

Proof. (i) $x(t) \sim x(t)$ since $x(b) = x(b)$
(ii) If $x(t) \sim y(t)$ then $y(t) \sim x(t)$ since $x(b) = y(b)$
(iii) If $x(t) \sim y(t)$ and $y(t) \sim z(t)$ then $x(t) \sim z(t)$ since $x(b) = y(b) = z(b)$

Proposition 7.4. *Let $\mathscr{B}$ be the set of all evaluation polynomials; then $\mathscr{B}/\sim$ is isomorphic to the ring of integers.*

Proof. Let base polynomials be the representatives of the equivalence classes of the set $\mathscr{B}$ with respect to the relation $\sim$. Since base polynomials are unique for all integers $x \in \mathbb{Z}$. The map

$$\begin{aligned} \mathbb{Z} &\to \mathscr{B}/\sim \\ x &\mapsto [(x, x(b))] \end{aligned}$$

gives the isomorphism.

Assume that $\mathbb{Z}_q$ is represented by the least residue classes $\mathscr{R} = \{0, 1, 2, \ldots, q-1\} \subset \mathbb{Z}$ (see Section 7.3.2). Evaluation polynomials defined on least residue set has a special importance for our terminology.

Definition 7.4. Let d be a positive integer. We define a **polynomial frame** as

$$\mathscr{B}_q^d = \{(y, y(t)) \in \mathscr{B} : \ \deg(y(t)) < d \text{ and } y_i \in \mathscr{R} \subset \mathbb{Z}\}$$

where y_i stand for the coefficients of y for $i = 0, 1, \ldots, d-1$.

Observe that

$$\mathbb{Z}_q \ncong \mathscr{B}_q^d/\sim$$

although it is correct to say $\mathscr{R}$ is equivalent to $\cong \mathscr{B}_q^d/\sim$ as a set.

On the other hand, if the frame $\mathscr{B}_q^d$ is considered, $\mathscr{B}_q^d$ is closed neither under the binary operations $\otimes$ nor $\oplus$. Thus, we remark that

$$\mathscr{B}_q^d \ncong \mathbb{Z}_q[t]/(t^d - 1).$$

However, there exists a one-to-one set map sending $(x, x(t)) \mapsto [x(t)]$ (recall that $[x(x)] = \{y(t) \in \mathbb{Z}[t] : x(t) \equiv y(t) \bmod t^d - 1\}$). Consequently, we take $\mathscr{B}_q^d$ as a simple subset of $\mathscr{B}$ without any structure on it.

7.2.2 Discrete Fourier Transform (DFT)

As computer arithmeticians, we started to build a terminology in the time domain; we represent signals by polynomials and put emphasis on their evaluations. In this section, we present the DFT as a map from the polynomial frames to a Fourier ring, and we start by introducing the Fourier rings.

Definition 7.5. Let R be a ring, the set $\mathscr{F}^d = \oplus_{i=0}^{d-1} R$ of ordered d-tuples

$$(X_0, X_1, \dots, X_{d-1})$$

where $X_i \in S$ forms a ring with componentwise addition and multiplication (also called direct sum of rings). For notation purposes, we denote these d-tuples with polynomials (i.e., $(X_1, X_2, \dots, X_d)$ will be written as $X_0 + X_1 t + \cdots + X_{d-1} t^{d-1}$). We named the ring $\mathscr{F}^d$ as the **Fourier ring** over R; moreover the elements are called **spectral polynomials** having **spectral coefficients**.

Remark 7.3. Throughout this text we consider only the Fourier rings over $\mathbb{Z}_q$. Therefore, we add the q subscript to our notation and denote the Fourier ring over $\mathbb{Z}_q$ by $\mathscr{F}_q^d$.

Now we can define the DFT map.

Definition 7.6. Assume that $\mathscr{B}_q^d$ is a polynomial frame and $\mathscr{F}_q^d$ is a Fourier ring over $\mathbb{Z}_q$. Let ω be a primitive d-th root of unity in $\mathbb{Z}_q$. The DFT map over $\mathbb{Z}_q$ is an invertible set map

$$\begin{aligned} DFT_d^{\omega} : \mathscr{B}_q^d &\to \mathscr{F}_q^d \\ (x, x(t)) &\mapsto X(t) \end{aligned}$$

defined as follows:

$$X_i = DFT_d^{\omega}(x(t)) := \sum_{j=0}^{d-1} x_j \omega^{ij} \bmod q \tag{7.1}$$

with the inverse

$$x_i = IDFT_d^{\omega}(X(t)) := d^{-1} \cdot \sum_{j=0}^{d-1} X_j \omega^{-ij} \bmod q \tag{7.2}$$

for $i = 0, 1, \dots, d-1$. Moreover, we write

$$x(t) \xleftrightarrow{DFT} X(t)$$

and say $x(t)$ and $X(t)$ are transform pairs where $x(t)$ is called a **time polynomial** and sometimes $X(t)$ is named as the **spectrum** of $x(t)$.

In the literature, DFT over a finite ring spectrum (7.1) is also known as the *Number Theoretical Transform (NTT)* . Moreover, if q has some special form such as a Mersenne or a Fermat number, the transform is named after this form i.e., *Mersenne Number Transform (MNT)* or *Fermat Number Transform (FNT)* .

Note that, unlike the DFT over the complex numbers, the existence of DFT over finite rings is not trivial. In fact, Pollard [2] mentions that the existence of primitive root d-th of unity and the inverse of d do not guarantee the existence of a DFT over a ring. He adds that a DFT exists in ring R if and only if each quotient field R/M (where M is maximal ideal) possesses a primitive root of unity. If $R = \mathbb{Z}_q$ is taken, one gets the following corollary:

Corollary 7.1. *There exists a d-point DFT over the ring $\mathbb{Z}_q$ that supports the circular convolution if and only if d divides $p-1$ for every prime p factor of q.*

Proof. We sketch the proof given in Chapter 6 of Blahut [3]. First, we cover the case where q is a prime power.

The converse is easier to prove. The DFT length d is invertible in $\mathbb{Z}_q$, if d and q are relatively prime (i.e., $dd^{-1} = 1 + kq$ for some k). Surely, any common factor of d and q must be a factor of 1, which is impossible. Moreover, any element ω having order d relatively prime to q has order that divides the Euler function $\phi(q) = (p-1)p^{m-1}$. Therefore, a d-point DFT does not exist in $\mathbb{Z}_q$ unless d divides $q-1$.

On the other hand, let p be an odd prime ($p = 2$ is trivial); then the non-units in $\mathbb{Z}_q$ form a cyclic group having order $\phi(q) = (p-1)p^{m-1}$. Let π be the generator of this group and $\omega = \pi^{bp^{m-1}}$ for any b dividing $p-1$. Since non-units in $\mathbb{Z}_q$ are cyclic, ω exists; all that remains is to show that the inverse DFT exists:

$$\begin{aligned} d^{-1}\cdot\sum_{j=0}^{d-1} X_j\omega^{-ij} \bmod q &= d^{-1}\cdot\sum_{j=0}^{d-1}\omega^{-ij}\sum_{j'=0}^{d-1} x'_j\omega^{-ij'} \bmod q \\ &= d^{-1}\cdot\sum_{j'=0}^{d-1} x'_j\sum_{j=0}^{d-1}\omega^{-i(j'-j)}. \end{aligned}$$

The sum on i is equal to d if $j' = j$, while if j' is not equal to j, then the geometric series summation becomes $(1-\omega^{-(j'-j)d})/(1-\omega^{-(j'-j)})$, which is zero since $j' - j \not\equiv 0 \pmod q$. Therefore,

$$d^{-1}\cdot\sum_{j=0}^{d-1}\omega^{-ij} \bmod q = d^{-1}\cdot\sum_{j=0}^{d-1} x_i(d\delta_{jj'}) \bmod q = x_i$$

as desired.

Now, let $q = p_1^{m_1}p_2^{m_2}\dots p_r^{m_r}$. The use of the Chinese remainder theorem guarantees the existence of a d-point DFT in $\mathbb{Z}_q$ if and only if d-point DFT exists in each factor ring, which is equivalent to, say, d divides $p_i - 1$ for all $i = 1,2,\dots,r$.

Example 7.2. In general, longer length DFTs are of utmost importance in many applications. Obviously, Corollary 7.1 states that DFTs over integer rings mostly suffer from short lengths. For instance, in the fairly large ring $\mathbb{Z}_{2^{31}+1}$, one can only define a 2-point transform since $2^{31}+1 = 3 \cdot 715827883$, though, in Section 7.4.2 we describe some solutions to overcome such problems.

7.2.3 Properties of DFT: Time–frequency dictionary

In the previous section, we relate the time and spectral polynomials by the DFT map; it is also possible to relate operations (formally we mean functions) in a similar manner. In other words, we relate a pair of maps ϕ and Φ defined on time and spectral polynomials respectively, if DFT map commutes with them. The next definition prepares a formal setup for this discussion.

Definition 7.7. Let ϕ and Φ be operations on time and spectral domains, respectively. We write

$$\phi \quad \xleftrightarrow{DFT} \quad \Phi$$

and say ϕ and Φ are transform pairs on $x(t)$ and sometimes declare that **the map** DFT_d^{ω} **respects the operation** ϕ on a point $x(t)$ if the following diagram commutes

$$\begin{tikzcd}
\mathscr{B}_q^d \arrow[r, "DFT"] \arrow[d, "\phi"] & \mathscr{F}_q^d \arrow[d, "\Phi"] \\
\mathscr{B} & \mathscr{F}_q^d \arrow[l, "IDFT"']
\end{tikzcd}$$

Equivalently, if the following equation is satisfied

$$\phi(x(t)) = IDFT_d^{\omega} \circ \Phi \circ DFT_d^{\omega}(x(t)). \tag{7.3}$$

Theorem 7.1. *(Fundamental) Let ϕ and Φ be operations on time and spectral domains respectively. The condition*

$$\phi(x(t)) \in \mathscr{B}_q^d$$

is necessary and sufficient for functions ϕ and Φ to be transform pairs on a point $x(t) \in \mathscr{B}_q^d$. We say an **overflow occurs** *for those cases in which $\phi(x(t)) \notin \mathscr{B}_q^d$.*

Proof. Let there exists a DFT map $DFT_d^{\omega} : \mathscr{B}_q^d \to \mathscr{F}_q^d$. By definition, $IDFT_d^{\omega} \circ \Phi \circ DFT_d^{\omega}(x(t))$ is an element of $\mathscr{B}_q^d$, hence $\phi(x(t))$ must be an element of $\mathscr{B}_q^d$.

In general, not having a nice domain, DFT does not globally commute with such function pairs. However, DFT respects various operations locally. Linearity, convolution and time–frequency shifting are some of these operations.

In the literature, such operations which are referred as the properties of DFT are essential for a better understanding of the nature of the transform. In fact, because of these properties, the Fourier transform becomes a powerful tool for applied sciences. We refer the reader to [4] for a general review of these properties.

In a finite ring setting, the existence conditions of these properties are quite different from a theory over complex numbers. In here, we present various properties and further state the existence conditions of the two most important, namely, linearity and convolution. We start with some notations:

Notation 1 *Let ω be a principal d-th root of unity, and $\Gamma(t)$ and $\Omega(t)$ be the spectral polynomials with coefficients consisting of negative and positive powers of ω respectively, as follows*

$$\begin{aligned}\Omega(t) &= 1+\omega^1 t+\omega^2 t^2+\ldots+\omega^{(d-1)}t^{(d-1)}\ ,\\ \Gamma(t) &= 1+\omega^{-1} t+\omega^{-2} t^2+\ldots+\omega^{-(d-1)}t^{(d-1)}\ .\end{aligned}$$

Notation 2 *Let $a \in \mathbb{Z}$ be a constant number, a degree d polynomial with all of its coefficients equal to a (i.e., $a(t) = a+at+at^2+\ldots+at^d$) is denoted by $a(t)$.*

Time and frequency shifts: Time and frequency shifts correspond to circular shifts when working with finite-length signals. Let $x(t) = x_0+x_1t+\ldots+x_{d-1}t^{d-1}$ and $X(t) = X_0+X_1t+\ldots+X_{d-1}t^{d-1}$ be a transform pair. The *one-term right circular shift* is defined as

$$\begin{tikzcd} x(t) \circlearrowright 1 := x_{d-1}+x_0t+\ldots+x_{d-2}t^{d-1} \arrow[d, leftrightarrow, "DFT"] \\ X(t) \odot \Omega(t) \end{tikzcd}$$

where $\odot$ stands for componentwise multiplication. Similarly, one performs the *one-term left circular shift* by multiplying the coefficients of $X(t)$ with negative power sequence of the principal d-th root of unity:

$$\begin{tikzcd} x(t) \circlearrowleft 1 := x_1+x_2t+\ldots+x_{d-1}t^{d-2}+x_0t^{d-1} \arrow[d, leftrightarrow, "DFT"] \\ X(t) \odot \Gamma(t) \end{tikzcd}$$

An arbitrary circular shift can be obtained by applying consecutive one-term shifts or using a proper ω power sequence. For instance, s-term circular left shift ($0 \le s \le d-1$) is achieved by multiplying $X(t)$ with $\Gamma_s(t) = 1+\omega^{-s}t+\omega^{-st}t^2+\ldots+\omega^{-s(d-1)}t^{d-1}$, componentwise.

Sum of sequence and first value: The sum of the coefficients of a time polynomial equals the zeroth coefficient of its spectral polynomial. Conversely, the sum of the spectrum coefficients equals d^{-1} times the zeroth coefficient of the time polynomial (i.e., $x_0 = d^{-1}\cdot\sum_{i=0}^{d-1} X_i\omega^{-i}$ and $X_0 = \sum_{i=0}^{d-1} x_i\omega^{i}$ as seen in Figure 7.1).

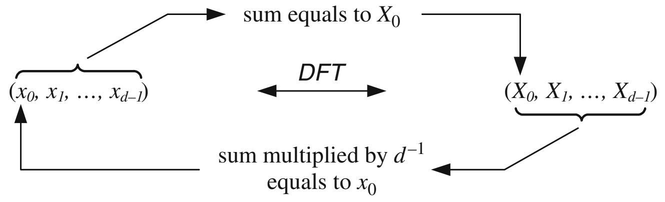

sum equals to X_0
DFT
$(x_0, x_1, \ldots, x_{d-1})$
$(X_0, X_1, \ldots, X_{d-1})$
sum multiplied by d^{-1} equals to x_0

Fig. 7.1 Sum of coefficients and first coefficient.

Left and right logical shifts. Using the above properties, it is possible to achieve the logical left and right digit shifts. We begin with the one-term left shift operation. Let $x_0(t)$ be equal to $x_0 + x_0 t + \ldots + x_0 t^{d-1}$ (see Notation 2) then

$$\begin{tikzcd} x(t) \ll 1 = (x(t) - x_0)/t = x_1 + x_2 t + \ldots + x_{d-1} t^{d-2} \arrow[d, leftrightarrow, "DFT"] \\ (X(t) - x_0(t)) \odot \Gamma(t) \end{tikzcd}$$

The right shifts are similar, where one then uses the $\Omega(t)$ polynomial instead of $\Gamma(t)$.

Using the fundamental Theorem 7.1, it is easily seen that time–frequency shifts and right/left shifts are globally respected by the DFT map. On the other hand, linearity and convolution properties are respected locally. We start by giving an example of overflow and then turn our attention back to state the conditions when these two properties are satisfied.

Example 7.3. Let ϕ_y be an operation such that

$$\phi_y : x(t) \mapsto x(t) + y(t) \text{ for all } x(t) \in \mathscr{B}_5^4$$

where $y(t) = 3 + t + t^2 + 4t^3$ is a base $b = 2$ evaluation polynomial for $y = 19$. Assume that the DFT map is a 4-point map over $\mathbb{Z}_5$, i.e., $DFT_4^{\omega} : \mathscr{B}_q^d \to \mathscr{F}_q^d$. Notice that the addition operation over the Fourier ring,

$$\Phi_y : X(t) \mapsto X(t) + Y(t) \text{ for all } X(t) \in \mathscr{F}_5^4$$

is a transform pair of ϕ_y on points $x(t)$ where

$$\phi_y(x(t)) = x(t) + y(t) \in \mathscr{B}_5^4 \subset \mathscr{B}. \tag{7.4}$$

Obviously, not all $x(t) \in \mathscr{B}_5^4$ satisfies Equation (7.4); for instance, if $x(t) = 3 + t^2 + t^3$ is an evaluation polynomial for $x = 15$, $\phi_y(x(t)) = x(t) + y(t) = 6 + t + 2t^2 + 5t^3$ gives an evaluation polynomial for 56 but

$$DFT_d^{\omega} \circ \Phi \circ DFT_d^{\omega}(x(t)) = 1 + t + 2t^2 \neq \phi_y(x(t)).$$

Therefore, we say ϕ_y and Φ_y are not transform pairs on $x(t) = 3 + t^2 + t^3$.

Observe that one gets the linearity property if the DFT map respects all the elements of the set $\{\phi_y : \text{for all } y(t) \in \mathscr{B}_q^d\}$ and its λ scaling for some $\lambda \in \mathbb{Z}_q$. Although the DFT map is a global group homomorphism over the additive group of complex numbers, it does not respect addition over finite-ring spectrum. The next proposition states that on a convex or a more regular subset of $\mathscr{B}_q^d$ the DFT map respects the single addition operation.

Proposition 7.5. *Let D be a subset of $\mathscr{B}_q^d$ such that $D = \{x(t) : x_i < q/2\}$ and ϕ_y be an addition operation on D for any $y \in D$. The DFT map respects ϕ_y on D.*

Proof. Let $\phi_y : D \mapsto \mathscr{F}_q^d$ be the addition map for any $y \in D$. Since $\phi_y(x(t)) = x(t) + y(t) \in \mathscr{B}_q^d$ for all $x(t)$ in D, using Theorem 7.1, DFT respects ϕ_y.

Next, we formally state when a DFT map respects the convolution operation. We start with a lemma:

Lemma 7.1. *Suppose that $(x(t), x)$ and $(y(t), y)$ are base b polynomials in the frame $\mathscr{B}_b^s$ with $s = \lceil d/2 \rceil$. The product $(z(t), z) = (x(t), x) \otimes (y(t), y)$ belongs to $\mathscr{B}_q^d$ where $q > sb^2$.*

Proof. Let $x(t) = x_0 + x_1 t + \ldots + x_{d-1} t^{d-1}$ and $y(t) = y_0 + y_1 t + \ldots + y_{d-1} t^{d-1}$ be polynomials such that $\deg(x(t)) + \deg(y(t)) < d$ and $0 \le x_i, y_i < b$ for some $b > 0$. Without loss of generality, assume $\deg(x(t)) \ge \deg(y(t))$. If $z(t) = x(t)y(t)$, then the coefficients of $z(t)$ can be written as follows:

$$z_k = \sum_{k=i+j} x_i y_j, \qquad k = 0, 1, 2, \ldots, d-1.$$

Notice that z_k can be found by adding at the most $\deg(y(t)) + 1$ nonzero terms, but since $\deg(y(t)) + 1 \le \lceil d/2 \rceil$, letting $s = \lceil d/2 \rceil$ gives

$$z_k \le (\deg(y(t)) + 1) \cdot b \cdot b \le s \cdot b^2$$

Thus, choosing $q > s \cdot b^2$ gives the result.

The following result gives the condition when the d-point DFT map respects the convolution of two elements of a frame $\mathscr{B}_q^d$.

Theorem 7.2. *Let $DFT_d^\omega : \mathscr{B}_q^d \to \mathscr{F}_q^d$ be a d-point DFT map and $D = \mathscr{B}_b^s$ be a subset of $\mathscr{B}_q^d$ such that $s = \lceil d/2 \rceil$ and $b^2 s < q$ for an integer $b > 0$. The DFT map respects the convolution map ϕ_y on $y(t)$ for any $y(t)$ in D where*

$$\phi_y : x(t) \mapsto x(t) \cdot y(t) \textit{ for all } x(t) \in D$$

Proof. Let $\phi_y : D \mapsto \mathscr{F}_q^d$ be the multiplication map for some $y(t) \in D$. By Lemma 7.1, the product $\phi_y(x(t)) = x(t) \cdot y(t) \in \mathscr{B}_q^d$ for all $x(t)$ in D. Therefore, using Theorem 7.1, DFT respects ϕ_y.

7.3 Spectral Modular Arithmetic

7.3.1 Time Simulations and Spectral Algorithms

In the previous section, we stated the conditions when the convolution and addition properties are respected by the DFT map over a finite ring spectrum. Since an algorithm is a combination of some primitive operations, starting with an example, we bring up the notion of algorithm pairs that are respected by DFT maps.

Example 7.4. Consider an algorithm in time domain performing the following operations:

```
Input: $x(t), y(t) \in \mathbb{Z}[t]$ polynomials of degree $d$
Output: $z(t) := x(t)(5y(t) + x(t)) - 3x(t)$
    1: $z(t) := x(t) + 5y(t)$
    2: $z(t) := x(t) \cdot z(t)$
    3: $z(t) := z(t) - 3x(t)$
    4: return $z(t)$
```

Whenever the DFT map respects the above algorithm, a dual algorithm operating in the spectrum can be furnished as follows:

```
Input: $X(t), Y(t) \in \mathbb{Z}[t]$ polynomials of degree $d$
Output: $Z(t) := X(t) \odot (5Y(t) + X(t)) - 3X(t)$
    1: $Z(t) := X(t) + 5Y(t)$
    2: $Z(t) := X(t) \odot Z(t)$
    3: $Z(t) := Z(t) - 3X(t)$
    4: return $Z(t)$
```

Once again we can relate these two objects using the DFT map and write

$$\text{Algorithm 1} \xleftrightarrow{DFT} \text{Algorithm 2}$$

Observe that when the inputs of Algorithm 7.4 and 7.4 agree, a parallel run produces the agreeing intermediate and final results. We name Algorithm 7.4 as the **time simulation** of the **spectral algorithm** (i.e., Algorithm 7.4).

Note that our primary interest in spectral techniques is to make use of the convolution property for calculating modular multiplications. An algorithm involving several multiplications benefits most from such a motivation. For instance, the encryption algorithm RSA [5] over some integer ring has such a nature but since these multiplications are modular, one has to deal with reductions. In the following sections, first, we describe a time simulation for modular reduction. Secondly, we translate the time simulation into a finite ring spectrum using the properties of DFT and finally, we analyze the minimal domains (i.e., smallest rings) in which our spectral algorithms work.

7.3.2 Modular Reduction

Before introducing the notion of spectral reduction, we need to make a few points clear about the modular arithmetic over the ring of integers;

In calculations of integers involving division it often happens that we are interested in the remainder, but not in the quotient. Those numbers having the same remainder when divided by a fixed number n are called congruent, to be more formal:

Definition 7.8. Let $n > 0$ be a fixed integer. We say x **is congruent to** y **modulo** n and write

$$y \equiv x \bmod n \quad \text{if } n \text{ divides } (y - x). \tag{7.5}$$

From the division algorithm we know that for each $x \in \mathbb{Z}$ there is an equation

$$x = nq + r, \qquad \text{for some } q \in \mathbb{Z} \text{ and } 0 \leq r \leq n$$

This means that each $x \in \mathbb{Z}$ can be assigned to one of the elements of the set $\{0, 1, 2, \ldots, n-1\}$. This set is called the **least residues** $\bmod n$ and it is clear that no two of the elements are congruent to each other $\bmod n$. We define the modular reduction as follows:

Definition 7.9. Let $n > 0$ be a fixed integer. We say y is the modular reduction of x modulo n and write

$$y = x \bmod n$$

where y is a least residue $\bmod n$.

Remark 7.4. The expressions "$y = x \bmod n$" and "$y \equiv x \bmod n$" have different meanings. Observe that the first one with "=" states that y is in the range $[0, n-1]$.

The equivalence on $\mathbb{Z}$ defined by the relation (7.5) partitions $\mathbb{Z}$ into n blocks, called the residue classes of $\mathbb{Z}$ modulo n. In fact, $\mathbb{Z}_n := \mathbb{Z}/n\mathbb{Z}$ is the set of these residue classes. If we denote the residue class modulo n containing y by $\bar{y}$, then the $\mathbb{Z}_n$ can be seen as the ring having the following n elements $\bar{0}, \bar{1}, \ldots, \overline{(n-1)}$. For instance, when $n = 2$, the residue classes are the set of even and odd numbers.

While performing computations such as modular exponentiation, in order to have some computational advantage, sometimes exact modular reduction calculations can be postponed for the intermediate values [6]. As long as these values belong to the correct residue classes, such modifications do not tend to misleading modular reductions. Now, we stretch the definition of the modular reduction for ease of our construction.

Definition 7.10. Let $\varepsilon > 0$ be an integer We call the set

$$F(\varepsilon) = \{y \in \mathbb{Z} : 0 \leq y < \varepsilon\}$$

the **integer frame of radius** ε.

Definition 7.11. Let $x, n > 0$ and $\varepsilon \geq n$ be integers. Then the elements of the set

$$\{y : y \in F(\varepsilon) \text{ and } y \equiv x \bmod n\}$$

are called as the **almost modular reductions of** x **with respect to the modulus** n.

Example 7.5. Let $\varepsilon = 12$ then $F(\varepsilon) = [0, 12)$ and the set of almost modular reductions of $x = 1$ with respect to modulus $n = 3$ is $\{1, 4, 7, 10\}$.

The choice of the radius ε completely depends on the nature or needs of the problem. Most of the time, the reductions are followed by a squaring or a multiplication. Therefore, as ε gets larger the operand sizes of the succeeding operations increase. Obviously $\varepsilon = n$ is the optimal choice in this sense. But, as we pointed out earlier, we are after some approximations of the optimal solution for some obvious reasons. In other words, we are looking for some small ε such that, after finding an element of almost modular reduction set, deducing the exact modular reduction has to be simple. Indeed, that is why it is appropriate to use the adjective "almost" to describe the elements of this set.

7.3.3 Spectral Modular Reduction

In this section, we give a formal definition for the spectral modular reduction and build up the necessary terminology for a better understanding of the algorithms in the spectrum. We return to our main objects: the set of evaluation polynomials, $\mathscr{B}$, and its subsets.

Proposition 7.6. *The evaluations of the polynomials in* $\mathscr{B}_r^d$ *form an integer frame* $F(\varepsilon)$ *in* $\mathbb{Z}$ *where* $\varepsilon = (r-1) + (r-1)b + (r-1)b^2 + \ldots + (r-1)b^{d-1}$.

Proof. It is easily seen that the polynomial $x(t) = (r-1) + (r-1)t + (r-1)t^2 + \ldots + (r-1)t^{d-1} \in \mathscr{B}_r^d$ attains the maximum evaluation value at base b which is the integer $(r-1) + (r-1)b + (r-1)b^2 + \ldots + (r-1)b^{d-1}$. The evaluation of the zero polynomial obviously gives the minimum value.

Definition 7.12. Let $n(t)$ be a base b polynomial of n with degree $d-1$ and $\mathscr{B}_r^d$ be a polynomial frame for some $r \geq b$. The elements of the set

$$\mathscr{A} = \{(y, y(t)) : y \equiv x \bmod n \text{ and } y(t) \in \mathscr{B}_r^d\}$$

are called **almost spectral reductions of the evaluation polynomial** $(x, x(t))$ **with respect to** $(n, n(t))$.

Lemma 7.2. *Let $\mathscr{A}$ be the set of all the almost spectral modular reductions of $(x, x(t))$ with respect to $(n, n(t))$. If $y(t)$ is the base polynomial for $y = x \bmod n$, then $(y, y(t)) \in \mathscr{A}$.*

Proof. If $y(t) = y_0 + y_1 t + \ldots + y_{d-1} t^{d-1}$ is the base polynomial for $y = x \bmod n$, then $0 \leq y_i < b$ for all $i = 0, 1, \ldots, d-1$. Since $r \geq b$, $y(t) \in \mathscr{B}_b^d \subset \mathscr{B}_r^d \Rightarrow (y, y(t)) \in \mathscr{A}$.

Definition 7.13. We call the base polynomial $y(t)$ of $y = x \bmod n$ as the **spectral (modular) reduction** of $(x, x(t))$ with respect to $(n, n(t))$ and we simply write

$$y(t) = x(t) \text{ mods } n(t).$$

Moreover the expression

$$y(t) \equiv x(t) \text{ mods } n(t)$$

mean n divides the evaluation of $(x(t) - y(t))$ at base b.

The spectral reduction defined in the time domain can be viewed as a projection of the usual modular operation in $\mathbb{Z}$ to the set of (evaluation) polynomials. Clearly, it is defined over the polynomials but it is different from the standard modular reduction in $\mathbb{Z}[t]$. To indicate this difference, in place of "mod" we choose to use "mods".

Similar objects can be defined for spectral polynomials; however, we note that unlike time polynomials, evaluation of spectral polynomials do not have any special meaning that serves our needs. To be specific, for a spectral polynomial $X(t)$, $X(b)$ does not have a special meaning, where $x(b)$ mostly represents a meaningful integer data. Therefore, our derivation for spectral polynomials is a notational continuation of the notation that is developed for time polynomials.

Definition 7.14. Let $x(t)$ be a base polynomial for $b > 0$ of an integer x. We call the spectral polynomial $X(t)$, the transform pair of $x(t)$, the **spectral base polynomial**.

Definition 7.15. Let $y(t)$ be an almost spectral reduction of $x(t)$ with respect to $n(t)$ in some frame $\mathscr{B}_r^d$. The spectral polynomial $Y(t)$, transform pair of $y(t)$, is called the **almost spectral reduction of** $X(t)$ **with respect to** $N(t)$, where $(X(t), x(t))$ and $(N(t), n(t))$ are transform pairs.

Definition 7.16. We call the base polynomial $Y(t)$ in the spectrum the **spectral modular reduction of** $X(t)$ **with respect to** $N(t)$ and we write

$$Y(t) = X(t) \text{ mods } N(t).$$

Moreover the expression

$$Y(t) \equiv X(t) \text{ mods } N(t)$$

means n divides the evaluation of $\mathrm{IDFT}((X(t) - Y(t))$ at base b.

7.3.4 Time Simulation of Spectral Modular Reduction

The spectral reduction can easily be achieved by deducing the base polynomial of the usual modular reduction (i.e., $y = x \bmod n$) but with such an approach one needs to perform classical modular reduction routines, which do not have simple spectral meanings. Our next step is to give a description of an algorithm that computes an almost spectral reduction of an evaluation polynomial.

Instead of a direct reduction method, here we present an algorithm of Montgomery type [7], which allows efficient implementation of modular arithmetic operations without explicitly carrying out the classical modular reductions. In fact, it replaces the modular reduction by a multiplication and some trivial shifts. The trick is, instead of attacking to compute "$x \bmod n$" directly, it proposes to derive it after performing a related computation

$$x \cdot \tau^{-1} \bmod n$$

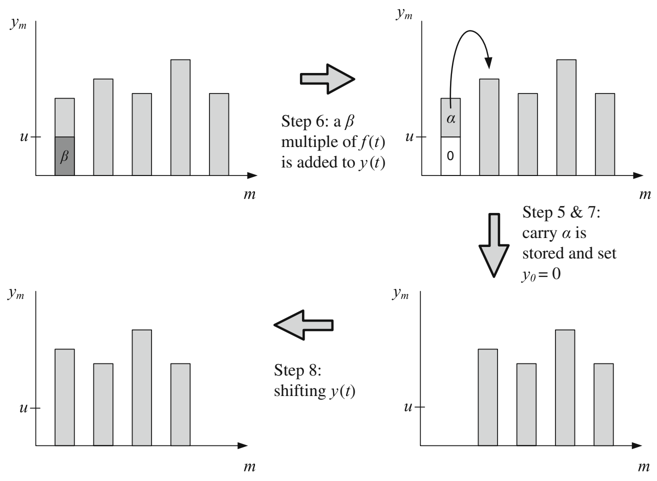
for $\tau > n$ and $\gcd(n, \tau) = 1$. At first glance, this seems computationally pointless because of the inversion involved but the selection of τ changes this first impression drastically. After giving some related notation, with Algorithm 7.3.4, we employ such a methodology.

Notation 3 *The polynomial product $x(t) \cdot t^e$ is denoted by $x^e(t)$, so in this context,*

$$x^{-e}(t) := x(t) \cdot t^{-e}$$

Time Simulation of Spectral Reduction Algorithm
Suppose that n and b are positive numbers with $\gcd(b,n) = 1$, $(n, n(t))$ is the base evaluation polynomial of degree $d-1$ and $(x, x(t)) \in \mathscr{B}_u^e$ where $e \geq d$ and $u \geq 0$.
Input: $x(t)$ and $n(t)$.
Output: $y(t)$, an almost spectral reduction of $x^{-e}(t)$ with respect to $n(t)$.

```
1:  Compute $\underline{n} = \delta n$ where $\underline{n}_0 = 1$ and $|\underline{n}_i| < b/2$
2:  $y(t) := x(t)$
3:  $\alpha := 0$
4:  for $i = 0$ to $e-1$
5:      $\beta := -(y_0 + \alpha)$ rem $b$
6:      $\alpha := (y_0 + \alpha + \beta)$ div $b$
7:      $y(t) := y(t) + \beta \cdot \underline{n}(t) \bmod q$
8:      $y(t) := (y(t) - y_0)/t$
9:  end for
10: $y(t) := y(t) + \alpha(t)$, for base polynomial $(\alpha, \alpha(t))$
11: return $y(t)$
```



y_m
u
β
m
Step 6: a β multiple of $f(t)$ is added to $y(t)$
α
0
Step 5 & 7: carry α is stored and set $y_0 = 0$
Step 8: shifting $y(t)$

Fig. 7.2 Illustration of the reduction of a single coefficient.

In Figure 7.2, we demonstrate the reduction of a single coefficient. Once the parameters α and β are calculated from the least significant coefficient, a β multiple of the modulus $\underline{n}(t)$ is added to the sum; carry passed to the next coefficient and finally the sum is shifted.

Algorithm 7.3.4 reduces the degree while reducing the radius (i.e., the magnitude of the coefficients). In fact, one can clearly compute the bound for the coefficients of the intermediate values. This gives us an opportunity to define a dual algorithm in the spectrum working properly with respect to a specific DFT map. We present these arguments formally in the next theorem.

Theorem 7.3. *Algorithm 7.3.4 computes an almost spectral reduction $y(t) \equiv x^{-e}(t)$ mods $n(t)$ such that the output signal $y(t) \in \mathscr{B}_r^d$ where $r = b^2 d + b$. Moreover, the coefficients of the intermediate values satisfy $0 \le y_i < u + b^2 d$ for $i = 0, 1, \ldots, d-1$.*

Proof. First of all, the algorithm computes an evaluation polynomial $y(t)$ such that $y(t) \equiv x(t) \cdot t^{-e}$ mods $n(t)$. This can be seen as follows: The value of $y(t)$ is accumulated either by adding a multiple of $n(t)$ or a term shift (i.e., Steps 7 and 8). Obviously, either adding a multiple of $n(t)$ or dividing when the least significant coefficient is zero does not change the residue class that $y(t)$ belongs to. Therefore, the result naturally follows because of shifting exactly e times.

Now, lets examine how big the coefficients get: by Definition (7.4), $(x, x(t)) \in \mathscr{B}_u^e$ implies that $0 \le x_i < u$ for all $i = 0, 1, \ldots, e-1$. Since $\deg(\underline{n}_d(t)) = d$ at every accumulation of the loop, $0 \le y_i < u$ for $i \ge d$. In particular, at the last d run $y_i = 0$ for $i > d$ and $0 \le y_d < b^2$.

On the other hand, when $i < d$ the coefficients of y_{d-i} satisfies

$$0 \le y_{d-i} < \beta \cdot \underline{n}_d (i+1) + u < b^2 (i+1) + u\ .$$

Observe that, a bound for y_{d-i} given by $0 \leq y_i < b^2 d + u$ for $i = 0, 1, \ldots, d-1$ is attained when $i = d-1$. The last d run is again special, since the reduction eliminates the u value from y_d at every accumulation. Therefore the final output satisfies

$$\begin{aligned} 0 \leq y_i &< (d-i)b^2 + b \\ &< r = b^2 d + b \quad \Longrightarrow \quad y(t) \in \mathscr{B}_r^d \end{aligned} \tag{7.6}$$

where the b factor comes from the last $\alpha(t)$ addition of Step 10 (note that, we assumed $deg(\alpha) < d-1$).

Note that Algorithm 7.3.4 describes a reduction routine of an arbitrary evaluation polynomial having degree $e-1$ larger than or equal to the degree of the modulus $d-1$. In fact, this does not really address the situation in a multiply–reduce methodology in which degrees are related. We give a corollary to Theorem 7.3 which cooperates with this situation.

Corollary 7.2. *Let $n(t)$ be a base b polynomial with degree $s-1$ such that $s = \lceil d/2 \rceil$ and let $x(t) \in \mathscr{B}_r^d$ where $r = sb^2$. Algorithm 7.3.4 computes an almost spectral reduction, $y(t) \equiv x(t) \cdot t^{-d}$ mods $n(t)$ in the polynomial frame $\mathscr{B}_{r'}^s$ where $r' = b^2 s + b$. Moreover, coefficients of all the intermediate values do not exceed $2b^2 s$.*

Proof. Let $n(t)$ be a base b polynomial with degree $s-1$ such that $s = \lceil d/2 \rceil$ and let $x(t) \in \mathscr{B}_r^d$ where $r = b^2 s$. Observe that the coefficients of $x(t)$ satisfy $0 \leq x_i < r = b^2 s$ for all $i = 0, 1, \ldots, d-1$ (note that we take $x(t)$ with the maximum degree $d-1$ in order to find the upper bounds). The algorithm drops the $\deg(x(t))$ to $\deg(n(t) = s-1$ and computes the almost spectral reduction of $x_{-d}(t) = x(t) \cdot t^{-d}$ in the frame $\mathscr{B}_{r'}^s$. The radius $r' = b^2 s + b$ can be deduced using Theorem 7.3. Moreover, since $0 \leq x_i < b^2 s$, the intermediate values are bounded by

$$0 \leq y_i < b^2 s + r = b^2 s + b^2 s = 2b^2 s.$$

7.3.5 Spectral Modular Reduction in a Finite Ring Spectrum

In this section, we translate the time simulation (i.e Algorithm 7.3.4) into the spectrum. We perform a line-by-line translation using the properties of DFT.

Our next step is to prove that Algorithm 7.3.4 and 7.3.5 agree; in other words there exists a DFT relation between the intermediate and output values in two domains at all times.

Theorem 7.4. *Algorithm 7.3.5 computes the almost spectral reduction, $Y(t) \equiv X^{-e}(t)$ mods $N(t)$ such that the inverse of the output signal $Y(t)$ gives $y(t) \equiv x^{-e}(t)$ mods $n(t)$ (i.e., the output of the Algorithm 7.3.4).*

Proof. Let $(x(t), X(t))$ and $(\underline{n}(t), \underline{N}(t))$ are transform pairs. In Step 4, we start with computing the last significant coefficient, y_0 of the time polynomial $y(t)$ using the

Spectral Reduction Algorithm (in a finite ring spectrum)
Suppose that there exists a DFT map $DFT_d^{\omega} : \mathscr{B}_q^e \to \mathscr{F}_q^e$ and

$$x(t) \xleftrightarrow{DFT} X(t), \qquad \underline{n}(t) \xleftrightarrow{DFT} \underline{N}(t)$$

where $(x(t),x) \in \mathscr{B}_r^e$ for $r < q - b^2 d$ and $(\underline{n}(t),\underline{n}) \in \mathscr{B}_b^{d+1}$ such that $\deg(n(t)) = d \le e$ and $\underline{n}$ is a multiple of modulus n with $\underline{n}_0 = 1$ (we assume $\gcd(b,n) = 1$).
Input: $X(t)$ and $\underline{N}(t)$, spectral polynomials
Output: $Y(t) \equiv X^{-e}(t)$ mods $N(t)$,

```
1:  $Y(t) := X(t)$
2:  $\alpha := 0$
3:  for $i = 0$ to $e-1$
4:      $y_0 := e^{-1} \cdot (Y_0 + Y_1 + \ldots + Y_d) \bmod q$
5:      $\beta := -(y_0 + \alpha) \bmod b$
6:      $\alpha := (y_0 + \alpha + \beta)$ div $b$
7:      $Y(t) := Y(t) + \beta \cdot \underline{N}(t) \bmod q$
8:      $Y(t) := Y(t) - (y_0 + \beta)(t) \bmod q$
9:      $Y(t) := Y(t) \odot \Gamma(t) \bmod q$
10: end for
11: $Y(t) := Y(t) + A(t)$,
12: return $Y(t)$
```

where $A(t)$ is the DFT pair of the base polynomial of α.

shifting property of DFT. Note that in Algorithm 7.3.4, y_0 comes for free. Once y_0 is computed, in Steps 5 and 6, the parameters β and the next carry α are generated.

In Step 7, a β multiple of $\underline{N}(t)$ is added to $Y(t)$, which updates $Y(t)$ such that $y_0 = 0 \bmod b$. By linearity, this step is equivalent to Step 6 of Algorithm 7.3.4.

Now, a division by t can be performed but before this shift, we need to eliminate the contribution of y_0 to the spectral polynomial $Y(t)$ completely. Since Step 7 updates y_0 to $y_0 + \beta$, the computation of $(Y(t) - (y_0 + \beta)(t))$ in Step 8 sets zeroth time term of $Y(t)$ to zero (observe that $(y_0 + \beta) \in \mathbb{Z}$ is a constant so $(y_0 + \beta)(t)$ is a fixed term polynomial, see Notation 2). If this is followed by the componentwise multiplication with $\Gamma(t)$ polynomial, Steps 8 and 9 together implement a logical circular shift (see Section 7.2.3).

Hence, we conclude that Algorithm 7.3.5 working in the spectrum agrees with Algorithm 7.3.4. However, we still need to find the domain for which these algorithms agree. We assume $\deg(x(t)) = e$ for $x(t) \in \mathscr{B}_r^e$ which implies that $0 \le x_i < r = (q - b^2 d)$ for $i = 0, 1, \ldots, e-1$. Since $\underline{n}$ is a multiple of modulus n with $\underline{n}_0 = 1$, we conclude by Theorem 7.3 that the intermediate values and the output $y(t)$ of the time simulation bounded by

$$0 \le y_i < r + b^2 d = q - b^2 d + b^2 d = q$$

Therefore, no overflows occur, Algorithms 7.3.4 and 7.3.5 generate the transform pair $y(t)$ and $Y(t)$. As Algorithm 7.3.4 computes $y(t) \equiv x^{-e}(t)$ mods $n(t)$, Algorithm 7.3.5 performs $Y(t) \equiv X^{-e}(t)$ mods $N(t)$.

With Algorithm 7.3.5 we have completed our primary discussion on spectral modular reduction. We leave the improvement-related comments to Section 7.4. Notice that our presentation so far targets the reduction of an arbitrary evaluation polynomial of degree e with respect to a base polynomial of degree $d < e$. In the next section, we change this routine and target to reduce an evaluation polynomial which is a result of a convolution. After this, we introduce the spectral modular multiplication.

7.3.6 Spectral Modular Multiplication (SMM)

Convolution and the SMR algorithm can easily be combined to harvest a spectral modular multiplication algorithm in a finite ring spectrum. In order to have a clear presentation we divide our presentation into 3 subprocedures as seen in Figure 7.3. Note that the initial and final stages consist of some data arrangements where the *Spectral Modular Product* (SMP) procedure consists of the actual multiplication and reduction steps (i.e., convolution and spectral reduction). Later, while presenting the spectral exponentiation algorithm, SMP is going to be the basic building block again. The SMP procedure and SMM are given as follows:

Since we operate in a finite ring spectrum, once again we need to deal with the overflows that might occur during the computations. In fact, Algorithm 7.3.6 gives a correct result if the intermediate values stay bounded. As a next step, we state the condition when overflows do not occur starting with two lemmas.

Lemma 7.3. *SMP*$(X^d(t), Y(t)) \equiv X(t) \odot Y(t)$ *mods* $N(t)$

Proof. Since SMP$(X^d(t), Y(t))$ computes the spectral coefficients of almost modular reduction, $Z(t) \equiv (X^d(t) \odot Y^{-d}(t))$ mods $N(t)$, hence taking the inverse transform gives

$$x^d(t)y^{-d}(t) = x(t) \cdot t^d y(t) \cdot t^{-d} = x(t) \cdot y(t) \text{ mods } n(t) \ .$$

Lemma 7.4. *Procedure 7.3.6 computes* $Z(t) \equiv (X(t) \odot Y^{-d}(t))$ *mods* $N(t)$ *if the parameters* b, q *and* s *satisfies the following inequality*

$$2sb^2 < q \tag{7.7}$$

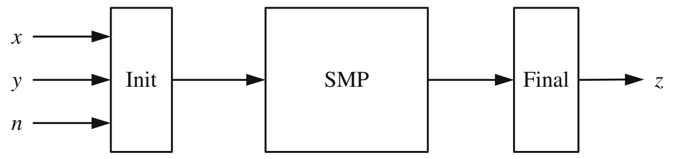


Fig. 7.3 Spectral Modular Multiplication.

Spectral Modular Product
Suppose that there exists a DFT map $DFT_d^{\omega} : \mathscr{B}_q^d \rightarrow \mathscr{F}_q^d$, and $X(t), Y(t)$ and $\underline{N}(t)$ be transform pairs of $x(t), y(t)$ and $\underline{n}(t)$ respectively where $(x(t), x)$ and $(y(t), y)$ are evaluation polynomials in the frame $\mathscr{B}_r^s$ with $r > 0$ and $s = \lceil d/2 \rceil$, and $(\underline{n}(t), \underline{n}) \in \mathscr{B}_b^{s+1}$ such that $\deg(\underline{n}(t)) \leq s$ and $\underline{n}$ is a multiple of modulus n with $\underline{n}_0 = 1$ (we assume $\gcd(b,n) = 1$).
Input: $X(t), Y(t)$ and $\underline{N}(t)$; spectral polynomials
Output: $Z(t) \equiv (X(t) \odot Y^{-d}(t))$ mods $N(t)$,
procedure SMP$(X(t), Y(t))$

```
1:  $Z(t) := X(t) \odot Y(t)$
2:  $\alpha := 0$
3:  for $i = 0$ to $d-1$
4:      $z_0 := d^{-1} \cdot (Z_0 + Z_1 + \ldots + Z_d)$ mod $q$
5:      $\beta := -(z_0 + \alpha)$ mod $b$
6:      $\alpha := (z_0 + \alpha + \beta)/b$
7:      $Z(t) := Z(t) + \beta \cdot \underline{N}(t)$ mod $q$
8:      $Z(t) := Z(t) - (z_0 + \beta)(t)$ mod $q$
9:      $Z(t) := Z(t) \odot \Gamma(t)$ mod $q$
10: end for
11: $Z(t) := Z(t) + \alpha(t)$
12: return $Z(t)$
```

Spectral Modular Multiplication
Suppose that there exists a DFT map $DFT_d^{\omega} : \mathscr{B}_q^d \rightarrow \mathscr{F}_q^d$. Let $n(t)$ be a base b polynomial for n where $\deg(n(t)) = s-1$, $s = \lceil d/2 \rceil$ and $\gcd(b,n) = 1$.
Input: A modulus $n > 0$ and $x, y < n$
Output: An almost modular reduction $z \equiv xy$ mod n

```
1:  Compute $\underline{n} = \delta \cdot n$ such that the base polynomial
    $\underline{n}(t)$ has degree $d$ and $\underline{n}_0 = 1$
2:  $\underline{N}(t) := \mathrm{DFT}_d^{\omega}(\underline{n}(t))$
3:  Compute the base polynomial $\lambda(t)$, $\lambda = b^d$ mod $n$.
4:  Compute the base polynomial $x^d(t) = x(t) \cdot t^d$
    for $x \cdot \lambda$ mod $n$.
5:  $X^d(t) := \mathrm{DFT}_d^{\omega}(x^d(t))$
6:  $Y(t) := \mathrm{DFT}_d^{\omega}(y(t))$
7:  $Z(t) := \mathrm{SMP}(X^d(t), Y(t))$
8:  $z(t) := \mathrm{IDFT}_d^{\omega}(Z(t))$
9:  return $z(b)$
```

Proof. In the previous sections, we described the action of the convolution and how the steps of reduction work. Here, we concentrate on driving the Inequality (7.7). Assuming that the conditions of SMP are satisfied, we investigate the time simulation of the algorithm in order to trace the overflows.

Using Theorem 7.3, observe that after convolution at Step 1, the time polynomial $z(t)$ doubles its degree to $2s-2$. Moreover, the magnitude of its coefficients cannot exceed sb^2 since $x(t)$ and $y(t)$ are base b polynomials (i.e., $z(t) \in \mathscr{B}_r^d$ where $r = sb^2$).

When it comes to analyzing the reduction steps, applying Corollary 7.2 assures that the output $z^{-d}(t)$ belongs to $\mathscr{B}^d_{r'}$ where $r' = b^2s + b$ and coefficients of all the intermediate values do not exceed $2b^2s$. Therefore, if q is chosen as $\max(2b^2s, b^2s + b) = 2b^2s < q$, no overflow is generated and SMP computes the desired result. Note here that the carry added in Step 12 is no longer large because of working with a convolution output, hence we take it as a constant rather than breaking it into words.

Theorem 7.5. *Algorithm 7.3.6 computes an almost modular reduction, $z \equiv xy \bmod n$, if the parameters b, q and s satisfy $2sb^2 < q$.*

Proof. Notice that in the initialization steps of Algorithm 7.3.6, before calculating the Fourier coefficients, we compute $xb^d \bmod n$ (i.e., $x^d(t) \bmod n(t)$). Using Lemma 7.3, one can see that Step 7 computes the product $x(t)y(t)$ mods $n(t)$ unless overflows occur.

Since the initialization and finalization parts do not change the coefficient bounds, one can get the minimal domain for the core SMP as $2sb^2 < q$ using Lemma 7.4.

7.3.7 Spectral Modular Exponentiation

In general, a single classical modular multiplication is faster than a single SMM; however, spectral methods are very effective when several modular multiplications with respect to the same modulus are needed. An example is the case when one needs to compute a modular exponentiation, i.e., the computation of $m^e \bmod n$, where m, e and n are positive integers. Such a setup needs a consecutive use of SMM; it also means a consecutive use of DFT and IDFT operations (obviously redundant computations as seen in Figure 7.4). Therefore, if the data is kept in the Fourier domain at all times, the backward and forward transforms are bypassed. Consequently,

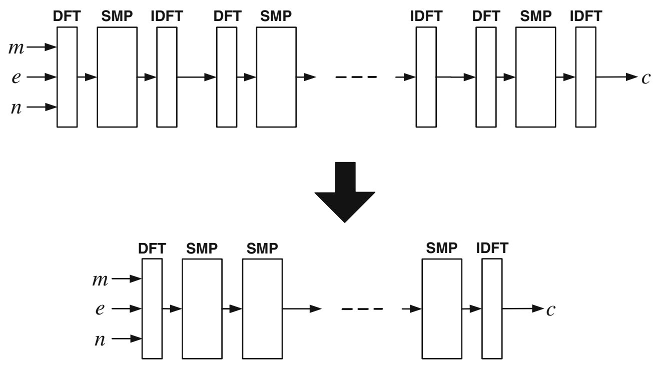


Fig. 7.4 Spectral Modular Exponentiation.

this approach decreases the asymptotic crossovers of the spectral methods to cryptographic sizes while computing the modular exponentiation.

There are many methods for carrying out general exponentiation. Mostly, efficiency comes from two resources; one is to decrease the time to multiply; the other is to reduce the number of multiplications. In practice one does both. Notice that, until now our objective was reducing the modular multiplication which is categorized in the first category. For the rest of this study, we keep this goal and simply consider using the binary method (see [8]) for the rest of our presentation.

The binary method scans the bits of the exponent either from left to right or from right to left. A squaring is performed at each step, and depending on the scanned bit value, a subsequent multiplication is performed. We describe the spectral modular exponentiation algorithm by using a left-to-right binary method below.

Remark 7.5. Since the SME algorithm computes an almost modular reduction of $c \equiv m^e \bmod n$, a final reduction may be needed if the output is desired in the range $[0, n-1]$.

Spectral Modular Exponentiation
Suppose that there exists a DFT map $DFT_d^{\omega} : \mathscr{B}_q^d \rightarrow \mathscr{F}_q^d$. Let $n(t)$ be a base b polynomial for n where $\deg(n(t)) = s-1$, $s = \lceil d/2 \rceil$ and $\gcd(b,n) = 1$.
Input: A modulus $n > 0$ and $m, e < n$
Output: An almost modular reduction, $c \equiv m^e \bmod n$.

```
1:  Compute $\underline{n} = \delta \cdot n$ such that the base polynomial
    $\underline{n}(t)$ has degree $d$ and $\underline{n}_0 = 1$
2:  $\underline{N}(t) := \mathrm{DFT}_d^{\omega}(\underline{n}(t))$
3:  Compute the base $b$ polynomial $(\lambda', \lambda'(t))$ where
    $\lambda' = b^{2d} \bmod n$.
4:  $\Lambda'(t) := \mathrm{DFT}_d^{\omega}(\lambda'(t))$
5:  $M(t) := \mathrm{DFT}_d^{\omega}(m(t))$
6:  $\overline{M}(t) := \mathrm{SMP}(M(t), \Lambda'(t))$
7:  $\overline{C}(t) := \mathrm{SMP}(1(t), \Lambda'(t))$
8:  for $i = j-2$ downto $0$
9:      $\overline{C}(t) := \mathrm{SMP}(\overline{C}(t), \overline{C}(t))$
10:     if $e_i = 1$ then $\overline{C}(t) := \mathrm{SMP}(\overline{C}(t), \overline{M}(t))$
11: $C(t) := \mathrm{SMP}(\overline{C}(t), 1(t))$
12: $c(t) := \mathrm{IDFT}_d^{\omega}(C(t))$
13: return $c(b)$
```

Once again, we need to guarantee that overflows do not occur; in other words, the coefficients of intermediate or final results should not be winding over q. We start with a lemma, stating how big the coefficients of a special polynomial get after a convolution, then, using this result we comment on how q has to be chosen to avoid overflows.

Lemma 7.5. *Let $s > 0$ and $x(t) = 1 + 2t + 3t^3 + \ldots + st^{s-1}$, then the coefficients of $z(t) = (x(t))^2$ are bounded by*

$$B(s) = \frac{-2s^3}{3} - s^2 + 2s^2 r_1 - \frac{s}{3} + \frac{r_1}{3} + 2sr_1 \tag{7.8}$$

where $r_1 = -2 + \sqrt{\frac{3+18s^2+18s}{9}}$.

Proof. Let $x(t) = 1 + 2t + 3t^3 + \ldots + st^{s-1}$ be a polynomial of degree $s-1$. Observe that if the convolution $z(t) = (x(t))^2$ is computed, the coefficients of $z(t)$ satisfies the following recurrence:

$$\begin{aligned}
z_0 &= 1 \\
z_1 &= 2^2 \\
z_2 &= 3^2 + z_0 \\
&\vdots \\
z_{s-1} &= s^2 + z_{s-3} \\
\\
z_s &= (s-1)(s+1) + z_{s-2} \\
z_{s+1} &= 2(s-2)(s+1) + z_{s-3} \\
&\vdots \\
z_{s+i-1} &= i(s-i)(s+1) + z_{s-i-1} \\
&\vdots \\
z_{2s-2} &= (s-1)(s+1) + z_0
\end{aligned}$$

If these coefficients are examined carefully one realizes that the coefficients up to the $(s-2)$th are monotonously increasing and $z_s > z_{s-2}$ for $s > 1$. Therefore, a maximum magnitude which also decides the bound has to be located somewhere in between coefficients $s-1$ and $2s-2$.

Observe that z_i is a telescoping sequence for $r < s$, in other words,

$$z_{r+1} + z_r = 1^2 + 2^2 + \ldots + (r+2)^2 = \sum_{i=1}^{r+2} i^2 \ .$$

This sum can be written as

$$z_{r+1} + z_r = \frac{(B + (r+2) + 1)^3 - B^3}{3} \tag{7.9}$$

where B^i stands for the ith Bernolli number (i.e., $B^0 = B_0 = 1, B_1 = -1/2, B_2 = 1/6$ and $B_3 = 1/30$. Plugging Bernolli numbers to the Equation (7.9) gives

$$z_{r+1} + z_r = \frac{1}{6}(2r^3 + 15r^2 + 37r + 30).$$

Here, if $r+1$ is even, then z_{r+1} and z_r can be found as

$$\begin{aligned} z_r &= z_1+z_3+\ldots+z_r = \frac{1}{6}(r^3+6r^2+11r+6) \\ z_{r+1} &= \frac{1}{6}(r^3+9r^2+26r+24) \end{aligned}$$

and, in case $r+1$ is odd, one would get

$$\begin{aligned} z_{r+1} &= z_1+z_3+\ldots+z_{r+1} = \frac{1}{6}(r^3+9r^2+26r+24) \\ z_r &= \frac{1}{6}(r^3+6r^2+11r+6). \end{aligned}$$

Therefore, in either case, z_r can be written as $\frac{1}{6}(r^3+6r^2+11r+6)$ and a general term of the recurrence would be found by plugging z_r into the system after replacing the index $s+i-1$ by r,

$$z_r = \begin{cases} \frac{1}{6}(r^3+6r^2+11r+6) & \text{if } r<s \\ -\frac{2s^3}{3}+s^2r+\frac{5s}{3}- & \\ \frac{r^3}{6}-\frac{11r}{6}+s^2+sr-r^2-1 & \text{if } s\le r<2s-1 \end{cases}$$

In order to find the maximum value of the z_r function, we substitute the roots of the derivative $\frac{\partial z_r}{\partial r}$ into the equation of z_r. Observe that the root $r_1 = -2+\sqrt{\frac{3+18s^2+18s}{9}}$ gives the local maximum in the range $s\le r<2s-1$ and if r_1 is plugged into z_r, one would get the bound $B(s)$ as a function of s

$$B(s) = \frac{-2s^3}{3}-s^2+2s^2r_1-\frac{s}{3}+\frac{r_1}{3}+2sr_1,$$

which in fact gives the desired bound.

Theorem 7.6. *Algorithm 7.3.7 computes $c\equiv m^e \bmod n$ if the parameters b,q and s satisfy the following inequality*

$$(b^2+b)^2B(s)+b^2s<q \tag{7.10}$$

where $B(s)$ is given by Equation (7.8).

Proof. First of all, SME implements the binary exponentiation method with a Montgomery type multiplier SMP all working in the spectrum. Thus, the algorithm works as long as a nice domain is chosen for all intermediate values and output causing no overflows. Recall by Theorem 7.3 that if the inputs of SMP are spectral base b polynomials, the output of SMP algorithm is a spectral polynomial having a time pair fitting into the frame $\mathscr{B}_r^s$ with $r=b^2s+b$. However, in the case of a consecutive SMP usage, the output of the second SMP would have larger time coefficients. For instance, in Steps 9, 10 and 11 the input $\overline{C}(t)$ is a spectral polynomial with time coefficients larger than b. If these steps are examined further, one understands that maximum magnitudes are attained from the computation of SMP$(\overline{C}(t), \overline{C}(t))$ in Step 9, since for both Steps 10 and 11, $\overline{M}(t)$ and $1(t)$ are spectral base b polynomials.

Now, we investigate how big the coefficients of the time polynomial get after Step 9. In order to have a better analysis, we look at the distribution of the coefficients of the time polynomial after applying SMP. In Theorem 7.3, we showed that after a reduction $c(t) \in \mathscr{B}_r^s$ has the form

$$c(t) = c_0 + c_1 t + c_2 t^2 + \ldots + c_{s-1} t^{s-1},$$

where $c_i < (s-i)b^2 + b$ for $i = 0, 1, \ldots, s-1$. Since $c_i < (s-i)b^2 + b < (s-i)(b^2 + b)$, we write

$$c(t) < (b^2 + b)y(t),$$

where $y(t) = s + (s-1)t + (s-2)t^2 + \ldots + 1t^{s-1}$. If $(c(t))^2$ is computed as seen in Step 9, we have

$$(c(t))^2 < (b^2 + b)^2 (y(t))^2,$$

and using Lemma 7.5, we guarantee that the coefficients of $(y(t))^2$ are bounded by $B(s)$ [1], which implies that $(b^2 s + b)^2 B(s)$ bounds the coefficients of $(c(t))^2$. When it comes to the intermediate values, because of the reduction steps, coefficients get slightly larger which is given by Theorem 7.3 as $(b^2 + b)^2 B(s) + b^2 s$. Therefore, if q is chosen larger than this final bound, no overflow is generated and the DFT respects SME over the ring $\mathbb{Z}_q$.

Inequality (7.10) is very centric as it gives the relation between the parameters b, s and q in a consecutive use of SMP algorithm. In practice, these parameters are generated in two different ways: the first one is picking s and b and then finding a suitable ring $\mathbb{Z}_q$ that admits a DFT of size d, while the second one is picking a ring with q elements, decide on s, and then find out the base b. We discuss the parameter selection-related issues in the next chapter after giving an illustrative example.

We conclude that $q > 131845.0 > 2^{17}$. Thus we need to search for a Fermat or Mersenne ring with $q \geq 2^{18} - 1$ that admits a DFT with length $d = 7$ or $d = 8$ for ω equal to a power of two. It turns out that the ring $\mathbb{Z}_{2^{20}+1}$ satisfies these conditions with $\omega = 32$.

7.3.8 Illustrative Example

In this section, we present the temporary values and the final result of an exponentiation computation (i.e., $c = m^e \pmod{n}$) using the SME method with the input values as $m = 2718$, $e = 53$, and $n = 3141$.

[1] $c(t)$ of Lemma 7.5 is the mirror image of $y(t)$

If we select the parameters $b = 2^3$ and $s = 4$, Inequality (7.10) assures that SME works correctly in a ring having $q > 131845$ elements. In order to have some computational convenience, we chose the Fermat ring $\mathbb{Z}_{2^{20}+1}$ which admits a DFT with length $d = 7$ for $\omega = 32$.

With these selections we compute $d^{-1} \bmod q$ and $\Gamma(t)$ as

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

and

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

We start with writing m and n in polynomial representation

$$\begin{aligned} n(t) &= 5 + 0 \cdot t + 1t^2 + 6t^3 \; , \\ m(t) &= 6 + 3t + 2t^2 + 5t^3 \; . \end{aligned}$$

Note that $\deg n(t) = s - 1 = 3$ and $\gcd(n, b) = 1$.

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

1. Given $n = 3141$, we have $n_0 = 5$. Finding the inverse of n_0 modulo b gives δ which is mostly achieved by Extended Euclidean algorithm:

   $$\delta = n_0^{-1} \bmod b = 5 \bmod 8 \; .$$

   Thus, $\underline{n} = \delta n = 5 \cdot 3141 = 15705$ which is equal to

   $$\underline{n}(t) = 1 + 3t + 5t^2 + 6t^3 + 3t^4$$

   in polynomial representation. Recall that $\deg(\underline{n}(t)) = s = 4$ and $\underline{n}_0 = 1$
2. The computation of $\underline{n}(t) = \mathrm{DFT}[\underline{n}(t)]$ can be accomplished by a matrix multiplication or, for more efficiency, some FFT can be employed. We obtain the result of the DFT as

   $$\underline{N}(t) = 18 + 201822t + 1045504t^2 + 93374t^3 + 856991t^5 + 3071t^6 + 944959t^7$$

   Recall that we work in the finite ring $\mathbb{Z}_q$ with $q = 2^{20} + 1 = 1048577$ represented by the least residue set; thus, the coefficients of the polynomial $\underline{N}(t)$ are in the range $[0, 2^{20})$.
3. After computing $\lambda' = 2^{2d} \bmod n = 8^{16} \bmod 3141 = 415$, the polynomial representation of λ' is found

   $$\lambda'(t) = 7 + 3t + 6t^2 \; .$$

Furthermore, we obtain the spectral coefficients of $\Lambda'(t)$ using the DFT as

$$\Lambda'(t) = 16+6247t+3073t^2+92167t^3+10t^4+6055t^5+1045506t^6+944136t^7$$

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

$$M(t) = 16+165990t+1046533t^2+96422t^3+886695t^5+2052t^6+948071t^7$$

5. The SMP algorithm is used to compute $\overline{M}(t) = \mathrm{SMP}[M(t),\Lambda'(t)]$ with inputs

$$\begin{aligned} M(t) &= 16+165990t+1046533t^2+96422t^3+886695t^5+2052t^6+948071t^7 \\ \Lambda'(t) &= 16+6247t+3073t^2+92167t^3+10t^4+6055t^5+1045506t^6+944136t^7 \end{aligned}$$

We then use the SMP to find the resulting polynomial $\overline{M}(t)$ given the inputs $M(t)$ and $\Lambda'(t)$. First we execute Step 1 in the SMP method, and obtain the initial value of $Z(t)$ using the rule $Z_i = M_i \cdot \Lambda'_i \bmod q$ for $i = 0,1,\ldots,7$ as

$$Z(t) = 256+945454t+10250t^2+236399t^3+223985t^5+1038347t^6+691376t^7$$

In Step 2 of the SMP method, we assign the initial value of $\alpha = 0$, and start the loop for $i = 0,1,\ldots,7$. We illustrate the computation of the instance of the loop for $i = 0$ in Table 7.1. The **for** loop needs to execute the remaining values of i as $i = 1,2,\ldots,7$ in order to compute the resulting product $\overline{M}(t)$ given by

$$\begin{aligned} \overline{M}(t) = {} & 135+324054t+36891t^2+398677t^3+27t^4+ \\ & 779927t^5+1011740t^6+594712t^7 \, . \end{aligned}$$

Table 7.1 The SMP **for** loop instance $i = 0$.

Step	Operation and Result
4:	$z_0 = d^{-1} \cdot (Z_0+Z_1+Z_2+Z_3+Z_4+Z_5+Z_6+Z_7) \pmod q$ $z_0 = 917505 \cdot (256+945454+10250+236399+223985+$ $1038347+691376) \pmod{1048567} = 42$
5:	$\beta = -(z_0+\alpha) \pmod b = -(42+0) \pmod{16} = 6$
6:	$\alpha = (z_0+0+\beta)/b = (42+6)/16 = 3$
7:	$Z_i = Z_i + \beta \cdot \overline{N}_i \pmod q$ $Z(t) = 364+59232t+1040389t^2+796643t^3+$ $123046t^5+8196t^6+69668t^7$
8:	$Z_i = Z_i - (z_0+\beta) = Z_i - 48 \pmod q$ $Z(t) = 316+59184t+1040341t^2+796595t^3+1048529t^4+$ $122998t^5+8148t^6+69620t^7$
9:	$Z_i = Z_i \cdot \Gamma_i \pmod q$ $Z(t) = 316+526138t+45048t^2+723385t^3+48t^4+$ $717053t^5+1003513t^6+130686t^7$

6. In this step, the SMP method is used to compute $\overline{C}(t) = \text{SMP}[1(t), \Lambda'(t)]$ with inputs

$$\begin{aligned} 1(t) &= 1 + t + t^2 + t^3 + t^4 + t^5 + t^6 + t^7 , \\ \Lambda'(t) &= 16 + 6247t + 3073t^2 + 92167t^3 + 10t^4 + 6055t^5 + 1045506t^6 + 944136t^7 \end{aligned}$$

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

$$\begin{aligned} \overline{C}(t) = {} & 106 + 13591t + 39979t^2 + 217142t^3 + 28t^4 + \\ & 11095t^5 + 1008684t^6 + 806969t^7 . \end{aligned}$$

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

$$\begin{aligned} \overline{M}(t) = {} & 135 + 324054t + 36891t^2 + 398677t^3 + 27t^4 + \\ & 779927t^5 + 1011740t^6 + 594712t^7 , \\ \overline{C}(t) = {} & 106 + 13591t + 39979t^2 + 217142t^3 + 28t^4 + \\ & 11095t^5 + 1008684t^6 + 806969t^7 . \end{aligned}$$

Given the exponent value $e = (53)_{10} = (110101)_2$, the exponentiation algorithm performs squarings and multiplications using the SMP 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 7.2. The final value is computed as

$$\begin{aligned} \overline{C}(t) = {} & 174 + 327348t + 43062t^2 + 592243t^3 + 54t^4 + \\ & 782837t^5 + 1005623t^6 + 395062t^7 . \end{aligned}$$

8. After the exponentiation loop is completed, we have the final value $\overline{C}(t)$. In this step, we have an SMP execution followed by an inverse DFT calculation.
9. We obtain $C(t)$ from $C(t) = \text{SMP}[\overline{C}(t), 1(t)]$ using the inputs

$$\begin{aligned} \overline{C}(t) = {} & 174 + 327348t + 43062t^2 + 592243t^3 + 54t^4 + \\ & 782837t^5 + 1005623t^6 + 395062t^7 . \\ 1(t) = {} & 1 + t + t^2 + t^3 + t^4 + t^5 + t^6 + t^7 . \end{aligned}$$

This computation finds $C(t)$ as

$$\begin{aligned} C(t) = {} & 169 + 438168t + 48142t^2 + 842167t^3 + 27t^4 + \\ & 696537t^5 + 1000463t^6 + 120506t^7 , \end{aligned}$$

Table 7.2 The steps of the exponentiation loop.

i	e_i	Operation	$\overline{C}(t)$
		Start	$106+13591t+39979t^2+217142t^3+$ $28t^4+11095t^5+1008684t^6+806969t^7$
5		$\overline{C}(t)$ =SMP$[\overline{C}(t),\overline{C}(t)]$	$127+13519t+36931t^2+118862t^3+$ $55t^4+11215t^5+1011780t^6+905297t^7$
	1	$\overline{C}(t)$ =SMP$[\overline{C}(t),\overline{M}(t)]$	$135+324054t+36891t^2+398677t^3+$ $27t^4+779927t^5+1011740t^6+594712t^7$
4		$\overline{C}(t)$ =SMP$[\overline{C}(t),\overline{C}(t)]$	$127+118020t+35890t^2+178339t^3+$ $45t^4+967557t^5+1012787t^6+833510t^7$
	1	$\overline{C}(t)$ =SMP$[\overline{C}(t),\overline{M}(t)]$	$175+434919t+42016t^2+648646t^3+$ $45t^4+693672t^5+1006625t^6+320201t^7$
3		$\overline{C}(t)$ =SMP$[\overline{C}(t),\overline{C}(t)]$	$119+526344t+16391t^2+982536t^3+$ $27t^4+589897t^5+1032200t^6+1047114t^7$
	0		$119+526344t+16391t^2+982536t^3+$ $27t^4+589897t^5+1032200t^6+1047114t^7$
2		$\overline{C}(t)$ =SMP$[\overline{C}(t),\overline{C}(t)]$	$202+128046t+60499t^2+955597t^3+$ $72t^4+976047t^5+988244t^6+37904t^7$
	1	$\overline{C}(t)$ =SMP$[\overline{C}(t),\overline{M}(t)]$	$192+628407t+33843t^2+586390t^3+$ $54t^4+494072t^5+1014836t^6+388633t^7$
1		$\overline{C}(t)$ =SMP$[\overline{C}(t),\overline{C}(t)]$	$265+755301t+60454t^2+460547t^3+$ $63t^4+422502t^5+988199t^6+459208t^7$
	0		$265+755301t+60454t^2+460547t^3+$ $63t^4+422502t^5+988199t^6+459208t^7$
0		$\overline{C}(t)$ =SMP$[\overline{C}(t),\overline{C}(t)]$	$296+546702t+74843t^2+734828t^3+$ $90t^4+606607t^5+973916t^6+209585t^7$
	1	$\overline{C}(t)$ =SMP$[\overline{C}(t),\overline{M}(t)]$	$174+327348t+43062t^2+592243t^3+$ $54t^4+782837t^5+1005623t^6+395062t^7$

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

$$c(t) = 56 + 59t + 42t^2 + 12t^3 \ .$$

Thus, the final value becomes $c(b) = 9360 \equiv 3078 \bmod 3141$, which is equal to

$$3078 = 22718^{53} \bmod 3141$$

as required.

7.4 Applications to Cryptography

Modular exponentiation is one of the most important arithmetic operation in modern cryptography. For example, the RSA algorithm requires exponentiation in $\mathbb{Z}_n$ for some positive integer n, whereas Diffie-Hellman key agreement and the ElGamal scheme use exponentiation in some large prime fields (see [9]).

In this chapter, we describe the methodologies of selecting the parameters for SME in order to use the method in public-key cryptography. We carefully investigate suitable rings and structures that makes spectral techniques available for modular arithmetic. In particular, the Inequality (7.10) presents a solid basis for the relation between the parameters b, q and s.

7.4.1 Mersenne and Fermat rings

An integer 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. Moreover, if the principal root of unity is chosen as a power of 2, spectral coefficients are computed only using additions and circular shifts. 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 7.3, we tabulate some suitable Fermat and Mersenne rings for SME function. Furthermore, we also tabulate a root of unity and the DFT length for each ring.

Observe that, it is possible to attain larger transform lengths; however, such a principal root of unity brings some further complexity to the computations. To be specific, multiplications with roots of unity involve additions as well as cyclic shifts. Some cases such as $\omega = \pm\sqrt{2}$ can be tolerable for longer transform sizes but other choices could be very costly. For instance, in $\mathbb{Z}_{2^{20}+1}$, $\omega = 4100$ is not a power of 2, hence every single multiplication with roots of unity is a 20-bit by 20-bit multiplication and not tolerable for our purposes.

Table 7.3 Parameters of NTT for $2^{16} < q < 2^{81}$.

ring $\mathbb{Z}_q$	prime factors	(ω, NTT length)	
$2^{16}+1$	65537	$(4,16)$	$(2,32)$
$2^{17}-1$	131071	$(2,17)$	$(-2,34)$
$2^{19}-1$	524287	$(2,19)$	$(-2,38)$
$2^{20}+1$	$17 \cdot 61681$	$(32,8)$	$(4100,16)$
$2^{23}-1$	$47 \cdot 178481$	$(2,23)$	$(-2,46)$
$2^{24}+1$	$97 \cdot 257 \cdot 673$	$(8,16)$	$(\sqrt{8},32)$
$2^{29}-1$	$233 \cdot 1103 \cdot 2089$	$(2,29)$	$(-2,58)$
$2^{31}-1$	2147483647	$(2,31)$	$(-2,62)$
$2^{32}+1$	$641 \cdot 6700417$	$(4,32)$	$(2,64)$
$2^{37}-1$	$223 \cdot 616318177$	$(2,37)$	$(-2,74)$
$2^{40}+1$	$257 \cdot 4278255361$	$(32,16)$	$(\sqrt{32},32)$
$2^{41}-1$	$13367 \cdot 164511353$	$(2,41)$	$(-2,82)$
$2^{64}+1$	$274177 \cdot 67280421310721$	$(4,64)$	$(2,128)$
$2^{79}-1$	$2687 \cdot 202029703 \cdot 1113491139767$	$(2,79)$	$(-2,158)$
$2^{80}+1$	$414721 \cdot 44479210368001$	$(32,32)$	$(\sqrt{32},64)$

In general, the transform lengths tabulated above are considered too short for most of the digital signal processing applications. On the other hand, these lengths seem reasonable for cryptographic applications and our purposes.

7.4.2 Pseudo Number Transforms

The Mersenne and Fermat rings are not the only suitable rings for efficient arithmetic, if m (not necessarily a prime) is a small divisor of n. The rings of the form $Z_{n/m}$ are also quite useful.

Definition 7.17. Let n and m be positive integers and m divides n. The NTT defined over $\mathbb{Z}_{n/m}$ is called a **pseudonumber transform (PNT)** .

In general, arithmetic in $\mathbb{Z}_{n/m}$ is difficult; however, since m is a factor of n, the arithmetic modulo (n/m) can be carried in the ring Z_n. By selecting Z_n as a Mersenne or Fermat ring one simplifies the overall arithmetic. The next theorem makes the importance of PNT more clear.

Theorem 7.7. *Let $n = n_1 n_2 \ldots n_l = m_1^{e_1} m_2^{e_2} \ldots m_l^{e_l}$, where $n_i = m_i^{e_i}$ for $i = 1, 2, \ldots, l$ for distinct primes m_i and positive integers e_i and l. Let R be a proper subset of the set $\{n_1, n_2, \ldots, n_l\}$ and $R' = \{m_i - 1 : m_i^{e_i} \in R\}$. If $S = \{m_1 - 1, m_2 - 1, \ldots, m_l - 1\}$, then gcd(S) $\leq$ gcd(R') $=: d'$ and a PNT of length-d' can be defined over $\mathbb{Z}_{n/m}$ for $m = \prod_{n_i \notin R} n_i$.*

Proof. First, $R \subsetneq S \Rightarrow \gcd(S) \leq \gcd(R')$. For the second part, let R be a proper subset of the set $\{n_1, n_2, \ldots, n_l\}$ such that $n/m = \prod_{n_i \in R} n_i$. Using Corollary 7.1, there exists an NTT with length $d' = \gcd(\{m_i - 1 : m_i^{e_i} \in R\})$ over $\mathbb{Z}_{n/m}$.

Example 7.6. In $\mathbb{Z}_{2^{15}-1}$, Corollary 7.1 states that the maximum transform length is $\gcd(6, 30, 150) = 6$. This MNT length is very short if the size of the ring is considered. On the other hand, if a PNT is employed in the ring $\mathbb{Z}_{(2^{15}-1)/7}$, we get the transform lengths up to $\gcd(30, 150) = 30$.

At first glance, the arithmetic in the ring $\mathbb{Z}_{(2^{15}-1)/7}$ seems difficult; however, it is possible to perform the actual computation in the ring $\mathbb{Z}_{(2^{15}-1)}$ with a final reduction to modulo $(2^{15} - 1)/7$.

Remark 7.6. Observe that PNT tailors the rings in a way that larger length transforms are possible. But while doing that, the size of the ring shrinks. The most interesting PNTs are the ones which enlarge the lengths with minimal shrinkage. The effective size of the decreased ring has to be concerned when PNTs are used.

In Table 7.4, we present parameters of some suitable pseudo-Mersenne and Fermat rings. If Tables 7.3 and 7.4 are combined, it is seen that for almost every v (recall that $n = 2^v \pm 1$) in between 16 and 41 there exists some sets of parameters for a nice NTT. Therefore, PNTs enrich the possible design choices which equip us to meet the marginal needs of particular applications.

Table 7.4 Suitable NTTs with ω and d values, $\star$ shows that n/m is a prime.

Ring n	PNT Modulus n/m	ω	d	ω	d
$2^{17}+1$	$(2^{17}+1)/3^\star$	$-2,4$	17	2	34
$2^{20}+1$	$(2^{20}+1)/17^\star$	4	20	2	40
$2^{23}-1$	$(2^{23}-1)/47^\star$	2	23	-2	46
$2^{23}+1$	$(2^{23}+1)/3^\star$	$-2,4$	23	2	46
$2^{25}-1$	$(2^{25}-1)/31$	2	25	-2	50
$2^{27}-1$	$(2^{27}-1)/511$	2	27	-2	54
$2^{28}+1$	$(2^{28}+1)/17^\star$	4	28	2	56
$2^{29}+1$	$(2^{29}+1)/3$	$-2,4$	29	2	58
$2^{31}+1$	$(2^{31}+1)/3^\star$	$-2,4$	31	2	62
$2^{34}-1$	$(2^{34}-1)/3$	2	34	-2	68
$2^{34}+1$	$(2^{34}+1)/5$	4	34	2	68
$2^{37}+1$	$(2^{37}+1)/3$	$-2,4$	37	2	74
$2^{39}-1$	$(2^{39}-1)/7$	2	39	-2	78
$2^{39}+1$	$(2^{39}+1)/9$	$-2,4$	39	2	78

7.4.3 Parameter Selection for RSA

In this section, we tabulate some SME parameters for modular exponentiation calculation suitable for RSA cryptosystems. Once the underlying ring, the DFT length and the principal root of unity are selected, the maximum modulus size used in the SME method is computed by finding the base $b = 2^u$. The relation between these parameters is computed after determining the maximum b satisfying the Inequality (7.10).

In Table 7.5, some sample rings with DFT parameters are given. We give an example to show how we get these figures. We first select a ring, for instance, let

Table 7.5 SMP parameter selection for SME.

Bits k	Ring $\mathbb{Z}_q$	DFT d	Root ω	Wordsize u	Words s
513	$(2^{57}-1)/7$	114	-2	9	57
518	$2^{73}-1$	73	2	14	37
704	$2^{64}+1$	128	2	11	64
1,185	$2^{79}-1$	158	-2	15	79
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
4,260	$(2^{142}+1)/5$	284	2	30	142

us take $q = 2^{79} - 1$. This comes with the principal root of unity $\omega = -2$, the length $d = 158$ and $s = \lceil d/2 \rceil = 79$. Plugging these values into the Inequality (7.10) gives

$$138754.3b^4 + 277508.7b^3 + 138833.3b^2 < 2^{79} - 1$$

and then, by inspection, $b = 2^{15} \Rightarrow u = 15$ is found. Therefore, we may perform an exponentiation of maximum operand size equal to $k = s \cdot u = 79 \cdot 15 = 1185$ using SME with the specified parameters.

7.4.4 Parameter Selection for ECC over Prime Fields

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

$$y^2 \equiv x^3 + ax + b \mod p \tag{7.11}$$

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 $(x_1, y_1) \in E(GF(p))$ and $(x_2, y_2) \in E(GF(p))$ 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

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

where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$.

Observe that all computations are performed within the finite field $GF(p)$. Therefore, spectral modular algorithms of the previous sections can be used for field operations. In particular, SMP can be used for multiplications.

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, e, which satisfies the equation

$$\mathbb{q} = e \times \mathbb{p} = \underbrace{\mathbb{p} + \mathbb{p} + \cdots + \mathbb{p}}_{e \text{ times}} \ ,$$

where $\mathbb{p}$ and $\mathbb{q}$ are points on the elliptic curve. Naturally, the basic computation (called *point multiplication*) in ECC is finding the eth (additive) power of an element $\mathbb{p}$ 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, $GF(p)$.

Therefore, the elliptic point multiplication operation can be performed using the SMP consecutively. Once again, Inequality (7.10) helps us to find parameters for ECC use. In Table 7.6, some sample rings with DFT parameters are given.

Table 7.6 SMP Parameter selection for ECC use.

Bits	Ring	DFT	Root	Wordsize	Words
k	$\mathbb{Z}_q$	d	ω	u	s
513	$(2^{57}-1)/7$	114	-2	9	57
518	$2^{73}-1$	73	2	14	37
704	$2^{64}+1$	128	2	11	64
1,185	$2^{79}-1$	158	-2	15	79
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
4,260	$(2^{142}+1)/5$	284	2	30	142

7.5 Spectral Extension Field Arithmetic

Spectral methods can be be applied to extension fields. Since binary and mid-size characteristic extensions are mostly of interest in practice, we breifly discuss these cases.

7.5.1 Binary Extension Fields

The most essential point of applying the spectral methods to binary field arithmetic is to find some suitable DFT domains having acceptable transform lengths for certain principal roots of unity. Unfortunately, if p is small, $\mathbb{Z}_p$ admits very short transform lengths (e.g., $\mathbb{Z}_2$ allows only a transform of length two). One way to overcome this problem is to use some polynomial rings over $\mathbb{Z}_p$ as the domain of DFT allows longer transform lengths because of their larger cardinality.

7.5.1.1 Suitable Polynomial Ring Spectrums

Spectral methods generally partition bigger problems into small pieces and then process the pieces in a parallel fashion. Notice that the computations in these pieces are carried in the ring, $R = \mathbb{Z}_2[\gamma]/(g(\gamma))$, hence for a proper $g(\gamma)$ selection, spectral methods benefit the most.

The most convenient choice of $g(\gamma)$ is a binomial. Moreover, if the principal root of unity, ω is chosen as a power of γ, the spectral coefficients are computed only using XORs and circular shifts. However, DFTs over polynomial rings having defining binomials suffer from the short transform lengths. For instance, $\gamma^n + 1$ has the linear factor $\Phi_1(\gamma) = \gamma + 1$ for all n, and by Theorem 1 of Pollard [2], only a transform length of two can be defined over these rings. Nevertheless, it is possible to overcome such restrictions using pseudotransforms (PT).

Pseudonumber transforms (PNT) are initially defined over subrings of Mersenne or Fermat rings. They support longer transform lengths and benefit the simplified arithmetic of the parent Mersenne or Fermat rings [10]. A similar approach can possibly be used for constructing pseudotransforms over polynomial rings. If $g(\gamma) = \gamma^n + 1$ is considered, a nice transform with a longer length can be grasped in a subring defined by a proper factor of $g(\gamma)$.

In general, factoring $\gamma^n + 1$ is not an easy problem which is closely related to the *cylotomic polynomials*. Since we are interested in binomials having fairly small degrees, even using a general purpose computer algebra system is satisfactory for our needs. Nevertheless, we present some pleasant arguments for the factors of cylotomic polynomials.

Definition 7.18. Let n be a positive integer, and let ω be primitive nth root of unity. The polynomial

$$\Phi_n(t) = \prod_{\gcd(n,k)=1} (t - \omega^n) \quad \text{for } 1 \leq k < n,$$

is called the nth cyclotomic polynomial .

The nth cyclotomic polynomial $\Phi_n(t)$ has degree $\varphi(n)$, where φ is the Euler's quotient function. These polynomials are irreducible over the rational numbers for every positive integer n but when they are considered over finite fields, this is no longer correct in general.

Since cyclotomic polynomials are minimal polynomials of the roots of unity, $g(\gamma) = \gamma^n \pm 1$ factor into cyclotomic polynomials. Consequently, the polynomial $\gamma^n - 1$ can be written as

$$\gamma^n - 1 = \prod_{d|n} \Phi_d(t).$$

Note that the above factorization is not necessarily prime over finite fields. For instance, $\gamma^5 - 1 = \phi_1(\gamma)\phi_4(\gamma)$ but $\phi_4(\gamma) = (t+1)^2$ over $GF(2)$. Letting p be an odd prime, some interesting examples over $GF(2)$ are as follows;

$$\begin{aligned}
t^p + 1 &= \Phi_1(t)\Phi_p(t), \\
t^{2p} + 1 &= \Phi_1(t)\Phi_2(t)\Phi_p(t)\Phi_{2p}(t), \\
t^{4p} + 1 &= \Phi_1(t)\Phi_2(t)\Phi_p(t)\Phi_{2p}(t)\Phi_{4p}(t), \\
&\vdots
\end{aligned}$$

One finds the first few remaining values of n as $t+1 = \Phi_1(t)$, $t^2+1 = \Phi_1(t)\Phi_2(t)$, $t^4+1 = \Phi_1(t)\Phi_2(t)\Phi_4(t)$, $t^8+1 = \Phi_1(t)\Phi_2(t)\Phi_4(t)\Phi_8(t)$ and $t^9+1 = \Phi_1(t)\Phi_3(t)\Phi_9(t)$.

Remark 7.7. In general, arithmetic in the factor rings is harder than the one in R, but being defined over a subring, PT calculations can be carried modulo $\gamma^n + 1$ for intermediate values and then the results are transformed to the factor ring by a final reduction. Such an approach simplifies the overall computation.

Example 7.7. Let us consider the DFT over $R = \mathbb{Z}_2[\gamma]/(\gamma^7+1)$ with the principal root of unity $\omega = \gamma$. Since γ^7+1 has the following factorization

$$\gamma^7+1 = \underbrace{(\gamma+1)}_{\Phi_1(\gamma)}\underbrace{(\gamma^3+\gamma^2+1)(\gamma^3+\gamma+1)}_{\Phi_7(\gamma)},$$

the ring R admits transform of lengths at the most two but if the ring $R' = \mathbb{Z}_2[\gamma]/(\Phi_7(\gamma))$, we get a 7-point DFT satisfying the convolution property over the ring R'. Besides that, one needs a $\Phi_7(\gamma)$ reduction while working in R' which is obviously harder than the arithmetic in R. However, since R' is a subring of R, all calculations can be carried over R with a final $\Phi_7(\gamma)$ whenever necessary.

In Table 7.7, we present the parameters for some suitable pseudotransform rings. One can find an appropriate $g(\gamma)$ by simply examining the transform length d in order to meet the marginal needs of a particular application.

Table 7.7 Suitable Polynomial Rings for an odd prime d.

Ring, $g(\gamma)$	ω	lenght
$(\gamma^d+1)/(\gamma+1), (\gamma^{2d}+1)/(\gamma^2+1)$	γ	d
$(\gamma^{d^2}+1)/(\gamma^d+1), (\gamma^{2d^2}+1)/(\gamma^{2d}+1)$	γ	d^2

Remark 7.8. While embedding the input to the pseudotransform domain, the size of the subring should be considered rather than the size of ring R. In fact, the most interesting pseudotransforms are the ones enlarging the lengths with minimal shrinkage in size. For further discussion we refer the reader to [11].

7.5.1.2 Suitable Finite Field Spectrums

We discuss the arithmetic simplifications when the factor rings are finite fields (i.e., defining polynomials are irreducible).

Note that binary extension fields can be seen as n-dimensional vector spaces over $GF(2)$: if $\{\alpha_1,\ldots,\alpha_n\}$ is taken as the basis set, each element of $GF(2^n)$ can be represented as a linear combination of the elements of this basis set. Among various bases, there are two special types having particular importance. The first one is the canonical polynomial basis $\{1,\alpha,\alpha^2,\ldots,\alpha^{n-1}\}$, made up of powers of a defining (mostly primitive) element α of $GF(2^n)$. The second one is the normal basis of the form

$$N = \{\alpha,\alpha^q,\ldots,\alpha^{q^{n-1}}\} \tag{7.12}$$

and consists of a normal element $\alpha \in GF(q^n)$ and its conjugates with respect to $GF(2)$.

For every finite field there exists a normal basis, in fact, several such bases may exist for the same field. Those bases having the minimal complexity while multiplying field elements are the most important ones for computations (also called optimal normal bases (ONB)).

For our purposes type I ONBs have the utmost importance in which the element α is taken as the principal root of unity. Observe that this is the case where the basis (7.12) and the set of roots of unity (i.e., $\{1, \omega, \omega^2, \dots, \omega^{n-1}\}$) become set equivalent (not necessarily equal as ordered sets); hence, one can change the basis from normal to polynomial or vice versa by simply ordering the terms. Unfortunately, not all the finite fields have type I ONB; the following proposition gives a condition for their existence.

Proposition 7.7. *Suppose $n+1$ is a prime and q is primitive in $\mathbb{Z}_{n+1}$, where q is a prime or prime power. Then the n nonunit $(n+1)$th roots of unity are linearly independent and they form an ONB of $GF(q^n)$ over $GF(q)$.*

Proof. See Mullin et al. [12]

Using the above result, one can get that for $k = 4, 10, 12, 18, 28, 36, 52, 58, 60, \dots$ the binary extension field $GF(2^k)$ has type I ONB.

In a normal basis representation, squaring a field element corresponds to a simple circular shift which seems well-suited for the realizations of public-key cryptosystems employing some form of repeated square and multiply methods, but in general, these representations mostly suffer from the complicated bases conversions and field multiplications. Eventually, type I ONB are optimal by giving the simplest conversion and multiplication realizations. Therefore, they initially favor a great interest in realizations of ECC but because of some security concerns, the use of elliptic curves over composite fields (type I ONB only exist in these extensions) is explicitly excluded from standards such as ANSI X9.63 [13].

Remark 7.9. We tend to choose a field having a type I ONB for transform domain. Observe that such a selection is implementation-related that does not change any ECC parameter, hence it never jeopardizes the security of the crypto- system.

7.5.1.3 Parameter Selection for ECC over Binary Fields

The size of the underlying structure (which also defines the key length) is a common security measure for public-key cryptosystems. After discarding the weak family of elliptic curves, standard documents [13] and [14] recommend some curves serving different needs of security levels. Referencing to the key sizes of these curves, in Table 7.8, we tabulate some suitable polynomial rings that admit nice DFT structures. Note that unlike SMM, when SMP is used for ECC, the word size $u \approx v/4$ as a result of successive SMP usage. In fact, a modification of SMP may give a much better u, (see the research project 1 of Section 7.8).

7.5.2 Midsize Characteristic Extension Fields

The ideas of the previous sections could be applied to the polynomial rings having midsize characteristics. In particular, extension fields $GF(q)$ with $q = p^k$, p an odd

Table 7.8 Standard Parameter Selection for SMP; † shows the domains having Type I ONB.

Bits	PT ring	DFT	Root	Wordsize	Words
k	$g(\gamma)$	d	w	u	s
171	$(\gamma^{37}+1)/(\gamma+1)$†	37	γ	9	19
210	$(\gamma^{41}+1)/(\gamma+1)$	41	γ	10	21
242	$(\gamma^{43}+1)/(\gamma+1)$	43	γ	11	22
288	$(\gamma^{47}+1)/(\gamma+1)$	47	γ	12	24
450	$(\gamma^{59}+1)/(\gamma+1)$†	59	γ	15	30
578	$(\gamma^{67}+1)/(\gamma+1)$†	67	γ	17	34

special prime, are good study cases. As we have seen in Section 7.5.1, when p is a small prime, DFTs suffer from short transform lengths. On the other hand, taking p such that $p \in [2^5, 2^{32}]$ gives great opportunities. Moreover, since the elements of the GF(q) could be represented by polynomials modulo p, one does not need to worry about the carries or coefficient overflows. In fact, this considerably simplifies SMP and any related computation. Let us start by giving the simplified SMP algorithm, and then we continue with further discussions.

7.5.2.1 Irreducible Binomials and Trinomials

As we mentioned in Section 7.5.1, the cryptosystems designed over extension fields give us the opportunity of choosing the parameter p and $f(t)$ freely. Certainly, we picked p as a Mersenne or Fermat prime or a large divisor of non-prime such numbers, and tend to choose $f(t)$ as a low hamming weight polynomial such as a binomial or a trinomial. Moreover, we insist on fixing the coefficients to powers of two, so that multiplications on the coefficients enjoy shifts instead of full multiplications.

We discussed the suitability of Mersenne and Fermat numbers earlier. Here, we start by giving the existence characterization of irreducible binomials. The next theorem is due to [15];

Spectral Modular Product

Assume that there exists a DFT map $DFT_d^{\omega} : \mathbb{Z}_p^d \rightarrow \mathscr{F}_p^d$, and $X(t), Y(t)$ and $F(t)$ are transform pairs of $x(t), y(t)$ and $f'(t)$ respectively, wherein, $x(t)$ and $y(t)$ are in the frame $\mathbb{Z}_p^s$ with $s = \lceil d/2 \rceil$, and $f'(t)$ is a multiple of the defining polynomial $f(t)$ in $\mathbb{Z}_p^{s+1}$ and $f'_0 = 1$.

Input: $X(t), Y(t)$ and $F(t)$; spectral polynomials

Output: $Z(t)$, spectral modular reduction of $x(t) \cdot y(t)$ mod $f(t)$,

procedure SMP$(X(t), Y(t))$

```
1:  $Z(t) := X(t) \odot Y(t)$
2:  for $i = 0$ to $d-1$
3:      $z_0 := d^{-1} \cdot (Z_0 + Z_1 + \ldots + Z_d) \bmod p$
4:      $Z(t) := Z(t) - z_0 \cdot F(t) \bmod p$
5:      $Z(t) := Z(t) \odot \Gamma(t) \bmod p$
6:  end for
7:  return $Z(t)$
```

Theorem 7.8. *Let $l \geq 2$ be an integer and $a \in GF^*(q)$. Then the binary polynomial $t^l - a$ is irreducible in $GF(q)[t]$ if and only if the following conditions are satisfied: (i) each prime factor of l divides the order e of a in $GF^*(q)$ but not $(q-1)/e$; (ii) $q \equiv 1 \bmod 4$ if $l \equiv 0 \bmod 4$.*

Proof. See pages 124–125 of [15].

As a corollary we specify the existence of irreducible binomials over Mersenne fields.

Corollary 7.3. *Let $q = 2^r - 1$ be a Mersenne prime and $\omega = \pm 2$, the binomials, $t^r - \omega^i$ are irreducible in $GF(q)[t]$ if and only if r^2 does not divide $q-1$ and gcd$(r,i) = 1$ for $i = 1, 2, \ldots, 2r$.*

Proof. We simply check whether the conditions of Theorem 7.8 are satisfied or not. We start with condition (ii); r has to be odd in order that q be a prime. Hence, r does not divide 4 and condition (ii) is always satisfied.

For condition (i), the order of ω^i surely divides the order of ω which is $|\omega| = r$ or $2r$. Since the set of roots of unity forms a cyclic subgroup of order r or $2r$, those elements with power relatively prime to r or $2r$ have order equal to r or $2r$; others are proper divisors.

As an example; by using Corollary 7.3, the irreducible binomials in $GF(q)[t]$ for $q = 2^{13} - 1$ that interest us most are given simply the form $t^{13} - 2^i$ for $i \neq 13$ and $i \in \{1, 2, \ldots, 25\}$.

When trinomials are considered, it is hard to characterize the conditions compactly. Therefore, once again we refer the reader to [15] for further reading. In fact, since we are interested in relatively small degree polynomials and these polynomials are comparably dense in $GF(p)[t]$, searching methods are suitable for finding such polynomials. In order to give some samples, we tabulate such polynomials in Table 7.9.

7.5.2.2 SMP with Binomials or Trinomials

If special irreducible binomials or trinomials are used for SMP algorithms, a significant improvement is possible. To be more specific, in Step 3 of SMP method we

Table 7.9 Some irreducible trinomials in $GF(q)[t]$ for $q = 2^{13} - 1$.

$t^{13}+t+2$	$t^{13}+t+2^{10}$	$t^{13}+t+2^{19}$
$t^{13}+t+2^2$	$t^{13}+t+2^{11}$	$t^{13}+t+2^{20}$
$t^{13}+t+2^3$	$t^{13}+t+2^{12}$	$t^{13}+t+2^{21}$
$t^{13}+t+2^4$	$t^{13}+t+2^{13}$	$t^{13}+t+2^{22}$
$t^{13}+t+2^5$	$t^{13}+t+2^{14}$	$t^{13}+t+2^{23}$
$t^{13}+t+2^6$	$t^{13}+t+2^{15}$	$t^{13}+t+2^{24}$
$t^{13}+t+2^7$	$t^{13}+t+2^{16}$	$t^{13}+t+2^{25}$
$t^{13}+t+2^8$	$t^{13}+t+2^{17}$	$t^{13}+t+2^{26}$
$t^{13}+t+2^9$	$t^{13}+t+2^{18}$	

subtract the z_0 multiple of $F(t)$ from the partial sum. If $f(t)$ is a random irreducible polynomial, this multiplication corresponds to a $v \times v$ multiplication but with the special trinomials or binomials this multiplication is performed by simple shifts.

Let $f(t) = t^m + \omega^{-s_0}$ with an integer s_0 be an irreducible binomial, $f'(t)$ simply equals $f'(t) = 1 + \omega^{s_0} t^m$. If the transform pair of $f'(t)$ is computed, one gets

$$F(t) = 1 + \omega^{s_0} + (1 + \omega^{s_0+m})t + \cdots + (1 + \omega^{s_0+m(d-1)})t^{d-1}.$$

Hence it is easily seen that the Step 3 of SMP follows

$$\begin{aligned} z_0 \cdot F_i &= z_0(1 + \omega^{s_0+mi}) \\ &= z_0 + z_0 \omega^{s_0+mi} \end{aligned}$$

for $i = 0, 1, \ldots, d-1$. Observe that all the $z_0 \omega^{s_0+mi}$ are computed by simple shifts because $\omega = 2$. Similarly, if $f(t)$ is a trinomial, another shift-add has to be performed.

7.5.3 Parameter Selection for ECC over Extension Fields

Spectral multiplication can be extremely efficient for extension fields having medium characteristics. By "medium" we mean the typical wordsize of today's architectures. For instance, if the field $GF(p^k)$ is considered we assume $2^7 < p < 2^{32}$.

In the literature, the security of an ECC employment is given according to the length of the key sizes. These key sizes are determined according to the complexities of the best-known algorithms known for solving the discrete logarithm problem in elliptic curve groups over the fields $GF(p^k)$. In Table 7.10, we tabulated the parameter selection of some nice Mersenne and Fermat fields that target some popular key sizes.

As mentioned in Section 7.4.2, psuedotransforms are also very convenient for employing spectral algorithms. In this context, if the prime p is chosen to be a

Table 7.10 Parameter Selection for ECC over $GF(p^k)$.

Bits	$GF(p^k)$	DFT	Root	Wordsize	Words
$s \cdot u$	p	d	ω	v	$s = k$
153	$2^{17}-1$	17	2	17	9
169	$2^{13}-1$	26	-2	13	13
190	$2^{19}-1$	19	2	19	10
256	$2^{16}+1$	32	2	16	16
289	$2^{17}-1$	34	-2	17	17
361	$2^{19}-1$	38	-2	19	19
496	$2^{31}-1$	31	2	31	16
512	$2^{16}+1$	64	$\sqrt{2}$	16	32
961	$2^{31}-1$	62	-2	31	31

Table 7.11 Parameter Selection for ECC over $GF(p^k)$ using PNTs.

Bits	$GF(p^k)$	DFT	Root	Wordsize	Words
$s \cdot u$	k	d	ω	u	s
150	$(2^{20}+1)/17$	20	4	15	10
170	$(2^{19}+1)/3$	19	4	17	10
255	$(2^{17}+1)/3$	34	-2	15	17
323	$(2^{19}+1)/3$	38	-2	17	19
352	$(2^{32}+1)/641$	32	4	22	16
464	$(2^{31}+1)/3$	31	4	29	16
483	$(2^{23}+1)/3$	46	2	21	23
644	$(2^{28}+1)/17$	56	2	14	23
899	$(2^{31}+1)/3$	62	2	29	31

divisor of the psuedo-Mersenne or Fermat number, one gets the parameters in Table 7.11 for efficient implementations. Although arithmetic modulo p might be difficult, the actual computation is carried in the chosen Mersenne or Fermat ring with a final modulo p reduction.

7.6 Notes

In this chapter new techniques of performing modular multiplication and exponentiation are proposed. Especially, modular exponentiation is one of the most important arithmetic operations for methods of modern cryptography, such as the RSA and Diffie-Hellman algorithms. The proposed methods use the Discrete Fourier Transform over finite rings, and relies on new techniques to perform the modular reduction operation.

The wonders of the convolution property has been known over decades. Obtaining modular arithmetic algorithms fully working in the spectrum would benefit the convolution property to the maximum extent. For carrying modular arithmetic, one need obviously has to deal with the concept of modular reduction. In [10] and later in a more compact text [16], after defining the spectral reduction and related concepts, a spectral reduction algorithm is introduced using the linearity and shifting property of DFT. Spectral modular multiplication (SMM) and spectral modular exponentiation (SME) come quite naturally once a reduction is defined.

When it comes to the practicability of the proposed methods, there were many directions to go because of the richness of the spectral theory. A first experiment could possibly work in a complex spectrum but, because of massive computations in the spectrum, the round-off errors could be hard to control (but still an analysis is needed). Therefore, a smart move is to employ the finite ring spectrums for not admitting the round-off errors in the computations. Additionally, from a computational point of view, calculations in some special rings such as Fermat and Mersenne can exploit the special arithmetic. In fact, there are excellent references [17], [18], [19] and [20] demonstrating efficient arithmetic in these rings. Moreover, one can check [4] for arithmetic in pseudotransform arithmetic.

The ideas utilized for modular operations of large integers are extendable to polynomial rings. The extension to the rings having mid-size characteristic is the easiest by simply using the underlying ring as the domain of the DFT. In fact, because of this simplicity, one does not need to worry about the carries or coefficient overflows and can have a very convenient method for the ring arithmetic. The method is independently proposed in [21] and [22]. Later a coprocessor [23] based on the method is introduced.

On the other hand, because of very short transform lengths (e.g., the ring $\mathbb{Z}_2$ allows only a transform two length of) using the spectral methods for binary or small characteristic ring extensions is a little cumbersome. One way to overcome this problem is to use some polynomial rings over $\mathbb{Z}_p$ as the domain of DFT allowing longer transform lengths [11] because of their larger cardinality.

Because of working in the spectrum, there exists a vast amount of parallelism potential in computations. Therefore, these methods have the chance of yielding efficient and highly parallel architectures especially for hardware implementations. Although we do not discuss implementation aspects in this text, the reader could find architectures and unit-gate analysis of the described methods in [10], [16], [11], [21] and [23].

7.7 Exercises

1. What is the maximum DFT lenght that can be defined over the ring $\mathbb{Z}_{2^{20}+1}$, $\mathbb{Z}_{2^{31}-1}$ and $\mathbb{Z}_{2^{79}-1}$? What will be this maximum if the principle root of unity is an integer power of two?
2. What are the best pseudotransform rings for $\mathbb{Z}_{2^{20}+1}$, $\mathbb{Z}_{2^{25}-1}$ and $\mathbb{Z}_{2^{39}+1}$ maximizing the DFT length? What will be these maximum lengths if the principle root of unity is an integer power of two?
3. What is the maximum modulus size that can be used for SME over the ring $\mathbb{Z}_{2^{20}+1}$? What will be this maximum if the principle root of unity is an integer power of two?
4. Assume that we want to use SME for an RSA system having a 1100 bits modulus. What is the smallest Mersenne and Fermat ring for the DFT such that SME works without overflows, if the principle root of unity is chosen as integer power of two.
5. Calculate $c = m^e \pmod{n}$ for $m = 2718$, $e = 53$, and $n = 3141$ using SME over the ring $\mathbb{Z}_{2^{19}-1}$.
6. What is the maximum DFT length (and relative the principal root of unity) that can be defined over the ring $R = \mathbb{Z}_2[\gamma]/(g(\gamma))$ where $g(\gamma) = (\gamma^{29}+1)$, $g(\gamma) = (\gamma^{29}+1)/(\gamma+1)$, $g(\gamma) = (\gamma^{49}+1)$ and $g(\gamma) = (\gamma^{49}+1)/(\gamma^7+1)$? What will be this maximum if the principal root of unity is power of γ?

7. What is the biggest binary field which its arithmetic can be carried using DFT over the ring $R = \mathbb{Z}_2[\gamma]/(g(\gamma))$, where $g(\gamma) = (\gamma^{19}+1)/(\gamma+1)$? What will be this field if the principal root of unity is a power of γ?
8. In characteristic three rings, DFT also suffers from short lengths. However, spectral methods can be applied similar to binary extensions. What is the maximum DFT length (and relative the principal root of unity) that can be defined over the ring $R = \mathbb{Z}_3[\gamma]/(g(\gamma))$, where $g(\gamma) = (\gamma^{23}+1)$, $g(\gamma) = (\gamma^{23}+1)/(\gamma+1)$, $g(\gamma) = (\gamma^{31}+1)$ and $g(\gamma) = (\gamma^{49}-1)/(\gamma^7-1)$? What will be this maximum if the principal root of unity is power of γ?
9. Assume that we want to employ a DFT for arithmetic in $GF(p^k)$ What is the maximum DFT length that can be defined over the ring $\mathbb{Z}_{2^{13}+1}$, $\mathbb{Z}_{2^{13}-1}$ and $\mathbb{Z}_{2^{79}-1}$? What will be this maximum if the principal root of unity is an integer power of two?
10. Let $m(t) = 6+3t+2t^2+5t^3 \in GF(p^k)$ for $p = 2^7-1$ and $k = 7$. If $f(t) = t^7-2$ is the defining polynomial for $GF(p^k)$ then calculate $c(t) = (m(t))^e \in GF(p^k)$ for $e = 53$.

7.8 Projects

1. If the SMP (i.e., Algorithm 7.5.2.1) is considered, notice that our bound analysis depends heavily on $\beta \cdot \underline{N}(t)$ multiplication of Step 7. In fact, it is possible to replace this multiplication by a multioperand addition at a cost of some precomputations and extra memory.
   To be more specific, let $b = 2^u$ and $n_i(t)$ be the polynomial representation of an integer multiple of n such that the zeroth coefficient of $n_i(t)$ satisfies $(n_i)_0 = 2^{i-1}$ for $i = 1,2,\ldots,u$ (note that $\underline{n}(t) = n_1(t)$). We can now write $\beta \cdot \underline{N}(t)$ as
   $$\beta \cdot \underline{N}(t) = \sum_{i=1}^{u} \beta_i \cdot N_i(t) , \tag{7.13}$$
   where β_i is a binary digit of β and $N_i(t) = \mathrm{DFT}_d^{\omega}(n_i(t))$ for $i = 1,2,\ldots,u$. Note that $\beta < b$ and $\beta_i = 0$ for $i \geq u$.
   Plugging the Equation (7.13) into the Algorithm 7.5.2.1 gives a modified spectral modular product (MSMP) algorithm. Observe the benefit of this approach since this replacement gives a reasonable amount of radius shrinkage. Calculate the new bound with respect to this modified algorithm.
2. In many situations it is desirable to break a congruence mod n into a system of small congruences modulo factors of n. Once necessary computations are performed in the small factor rings, using CRT , the resultant system of congruences is replaced by a single congruence under certain conditions.
   When spectral algorithms are considered, CRT can be used in two different ways. The first one is for degree that can be adopted from Quisquater and Couvreur [24] where the second one is for radius that is based on the ideas proposed for integer

multiplication by J. M. Pollard [25], and independently by A. Schönhage and V. Strassen [1].
Examine these two utilizations and present algorithms for both methods. Give a boundary anaylsis for both of the algorithms. Give a parameter selection table for popular RSA sizes using SMP and MSMP.

3. Note that Mersenne arithmetic corresponds to one's complement arithmetic whereas Fermat arithmetic may be implemented in different fashions. Research on efficient Fermat ring arithmetic ([20] and [17] could be two decent starting articles), then design parallel and digit serial Mersenne and Fermat multipliers for $\mathbb{Z}_{2^{13}-1}$ and $\mathbb{Z}_{2^{24}+1}$ respectively. Plot the relation between the area and digit size.
4. Consider the Fermat ring $\mathbb{Z}_{2^{24}+1}$. If $\omega = 8$ is taken, one gets a DFT having length 16. Calculate the parameters u and b in order to determine the maximum supported RSA length. Design a hardware architecture for SMP over the Fermat ring $\mathbb{Z}_{2^{24}+1}$.
5. Consider the Mersenne ring $\mathbb{Z}_{2^{13}-1}$. If $\omega = -2$ is taken, one gets a DFT having length 26. Design a an SMP architecture over the ring $\mathbb{Z}_{2^{13}-1}$ performing $GF(p^k)$ arithmetic for $p = 2^{13} - 1$, $k = 13$ and $f(t) = t^{13} - 2$ is the defining polynomial.

References

1. A. Schönhage and V. Strassen. Schnelle multiplikation grosser zahlen. *Computing*, 7: 281–292, 1971.
2. J. M. Pollard. Implementation of number theoretic transform. *Electronics Letters*, 12(15): 378–379, July 1976.
3. R. E. Blahut. *Fast Algorithms for Digital Signal Processing*, Addison-Wesley publishing Company, 1985.
4. H. J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*, Springer, Berlin, Germany, 1982.
5. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2): 120–126, February 1978.
6. T. Yanık, E. Savaş, and Ç. K. Koç. Incomplete reduction in modular arithmetic. *IEE Proceedings – Computers and Digital Techniques*, 149(2): 46–52, March 2002.
7. P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170): 519–521, April 1985.
8. Ç. K. Koç. High-Speed RSA Implementation. Tech. Rep. TR 201, RSA Laboratories, 73 pp. November 1994.
9. N. Koblitz. *A Course in Number Theory and Cryptography*, Springer, Berlin, Germany, Second edition, 1994.
10. G. Saldamlı. *Spectral Modular Arithmetic*, Ph.D. thesis, Department of Electrical and Computer Engineering, Oregon State University, May 2005.

11. G. Saldamlı and Ç. K. Koç. Spectral modular arithmetic for binary extension fields. *preprint*, 2006.
12. S. A. Vanstone, R. C. Mullin, I. M. Onyszchuk and R. M. Wilson. Optimal normal bases in GF(p^k). *Discrete Applied Mathematics*, 22: 149–161, 1989.
13. ANSI X9.62-2001. Public-key cryptography for the financial services industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography. 2001, Draft Version.
14. IEEE. *P1363: Standard specifications for public-key cryptography*. November 12, 1999, Draft Version 13.
15. R. Lidl and H. Niederreiter. *Finite Fields*, Encyclopedia of Mathematics and its Applications, Volume 20. Addison-Wesley publishing Company, 1983.
16. G. Saldamlı and Ç. K. Koç. Spectral modular arithmetic. In *Proceedings of the 18th IEEE Symposium on Computer Arithmetic 2007 (ARITH'07)*, 2007, pp. 123–132.
17. J.-L. Beuchat. A family of modulo $(2^n + 1)$ multipliers, *Tech. Rep. 5316*, Institut National de Recherche en Informatique et en Automatique (INRA), September 2004.
18. Z. Wang, G. A. Jullien, and W. C. Miller. An efficient tree architecture for modulo $2^n + 1$ multiplication. *J. VLSI Signal Processing Systems*, 14(3): 241–248, December 1996.
19. R. Zimmermann. "Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication," in *Proceedings of the 14th IEEE Symposium on Computer Architecture*, 1999, pp. 158–167.
20. L. M. Leibowitz. A simplified binary arithmetic for the Fermat number transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24: 356–359, 1976.
21. G. Saldamlı and Ç. K. Koç. Spectral modular arithmetic for polynomial rings. *preprint*, 2006.
22. S. Baktir and B. Sunar. Finite field polynomial multiplication in the frequency domain with application to Elliptic Curve Cryptography. In *Proceedings of Computer and Information Sciences ISCIS 2006)*, pp. 991–1001, 2006.
23. S. Baktir, S. Kumar, C. Paar, and B. Sunar. A state-of-the-art elliptic curve cryptographic processor operating in the frequency domain. *Mobile Networks and Applications (MONET)*, 12(4): 259–270, September 2007.
24. J.-J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronics Letters*, 18(21): 905–907, Oct. 1982.
25. J. M. Pollard. The fast Fourier transform in a finite field. *Mathematics of Computation*, 25: 365–374, 1971.